Final 2.0

ITAA GEIA Group

FAA SWIM Program

Segment 1 to Segment 2 Transition – Industry Input

12 December 2008

Acknowledgements

The Federal Aviation Administration has requested industry input on the transition from SWIM Segment 1 to SWIM Segment 2 with regards to the use Service Oriented Architecture (SOA) for the FAA SWIM program. The Information Technology Association of America's GEIA Group includes many industry partners who support the FAA, and ITAA/GEIA formed a working group to prepare this whitepaper in response to the FAA request.

Thanks to the following people for contributing to this whitepaper:

Doc. Version: Final 2.0
Editor: D. Sweigard
Contributors:

Organization	Participant	Organization	Participant
Harris Corp.	David Almeida	Iona	Bob Kilker
Tectura	Bill Barr	IBM	Mike Moomaw
Oracle	Peter Bostrom	Boeing	John Moore
Boeing	Mike Kaufman	ARINC	Andrew Onken
SITA	Kathleen Kearns	CSC	Jay Pollack
Iona	Michelle Davis	Boeing	Les Robinson
Harris Corp.	John Dockendorf	Harris Corp	Tom Schabowsky
Raytheon	Tim Donovan	Lockheed Martin	Al Secen
L-3 Communications	Chris Francis	Boeing	Al Sipe
Tectura	Kevin Halligan	Boeing	Bob Stephens
ARINC	Eric Harrell	Lockheed Martin	Doug Sweigard
Boeing	Gary Jackson	McAfee (formerly Secure Computing)	Elan Winkler
McAfee (formerly Secure Computing)	Lee Copeland	SITA	Mansour Rezaie-Mazinani
Pegasus Management, Inc.	Guinn Clark		

About the GEIA Group

The GEIA Group within ITAA develops and distributes forecasts of the Federal marketplace, creates best-practice industry standards, and maintains a committee structure through which its member companies work with representatives of FAA and other Federal agencies on matters of mutual concern. Previously a separate organization, in 2008 GEIA merged with the Information Technology Association of America (ITAA) and assumed the ITAA name. GEIA Group contact:

Dan C. Heinemeier, CAE
Executive Vice President, GEIA Group
Information Technology Association of America
1401 Wilson Boulevard, Arlington, VA 22209
www.geia.org/www.itaa.org
703-907-7565 · danh@itaa.org

Table of Contents

List of Figures

List of Tables

1 Executive Summary

Charter: In April of 2008, FAA requested that ITAA-GEIA convene an industry working group of its constituent members. The purpose of this working group was to write an industry white paper titled "SWIM Segment 1 to Segment 2 Transition". ITAA-GEIA solicited volunteers from their membership to participate in this endeavor resulting in a team composed of representatives from fourteen (14) different companies. A kick-off meeting between the FAA and this working group was held on June 3, 2008 to discuss the FAA's objectives for this effort. At the kick-off meeting, the ITAA-GEIA working group was chartered to provide, through industry best practices, alternatives for transitioning from the current SWIM Segment 1 architecture and deployment model to a proposed Segment 2 model.

Methodology: The working group utilized a team consensus approach in the development of this white paper. Weekly teleconferences were conducted with the team to ensure adequate communication and collaboration was occurring within the team. The team collaboratively developed an outline for the document and presented it to the FAA for concurrence, to ensure it was headed in the direction the FAA desired. Volunteers were solicited from within the team to lead the writing of the different sections of the paper. The authors collaborated with other team members outside of the weekly meetings as needed and relevant issues were the subject of discussion during the weekly meetings. For specific sections, such as section 2.3 "Discussion of What Segment 2 Should Be" (from a SOA) perspective", we formed a small working group to develop the concepts for the section and then reviewed it with the entire team at one of the weekly meetings. Once consensus was obtained for the approach authors were assigned to provide the detailed written material. All team members were asked to review the document and provided comments to ensure that a true consensus among the team members was attained.

Structure: The first part of Section 2 presents a brief description of the SWIM Segment 1 architecture to provide context for the Segment 2 recommendation. The remainder of Section 2 provides a discussion of the architectural alternatives that were considered, issues associated with each alternative, a list of selection criteria that was used by the team to assess each of the alternatives, the team's recommendation for the SWIM Segment 2 SOA architecture and provides a brief Operations Concept. Section 3 of the documents provides a detailed look at SOA concepts, providing a more academic type discussion of the potential structure of Segment 2 and relevant issues to be considered along the way. Section 4 of the documents focuses on Transition and Migration concerns for the implementation of a SOA infrastructure that are relevant to all segments of the SWIM program. Section 5 provides the team's initial pass at the possible risks that need to be addressed which the FAA can use to augment their existing SWIM program risk registry. In some cases potential risk mitigation strategies are also provided. Section 6 provides a list of recommendations for additional analysis and studies that the team thinks FAA should consider as work for Segment 2 progresses.

Recommendation: This ITAA-GEIA working group recommends that the FAA consider a Shared Services type of SOA architecture that will be able to support SWIM Segment 2 and beyond. This approach leverages existing infrastructure plus features flexibility and scalability advantages that will allow the FAA to grow and tailor the SOA infrastructure to meet their evolving business needs without forcing the demands of the other architectural alternatives on all NAS stakeholders.

2 Introduction/Overview

2.1 Current Segment 1 Description of Infrastructure Requirements

SWIM is both an architectural approach to integrating software components and a program within the FAA to realize the approach. It embraces a Service-Oriented Architecture (SOA) which separates functions into distinct units accessible over a network to support their combination and reuse. Applications offer services to others via an enterprise interface rather than multiple point-to-point interfaces. SWIM provides common standards, tools, and infrastructure to allow developers to focus on application logic, rather than integration mechanisms. Infrastructure in this context consists of both the logical and physical elements that facilitate system interaction.

The SWIM Segment 1 infrastructure requirements are presented in the *SWIM Service Specification Document (SvSD) Segment 1*. Those requirements are traceable to the top-level requirements defined in the *SWIM Segment 1 Final Program Requirements (FPR).*
The SWIM infrastructure is provided collectively by four Core Service: Interface Management, Messaging, Enterprise Service Management (ESM), and Security.

2.1.1 Interface Management

Interface Management provides standards and a core service to providers and consumers for interface definition, description, and discovery. In general, interface management functionality applies at both design-time and run-time. SWIM Segment 1 program objectives only entail design time functionality.

2.1.2 Messaging

Messaging includes functions supporting operation and data exchanges with a variety of relationships between message end-points, including one-to-one and one-to-many. It enables message routing and the distribution of content, as well as functions for efficiently and reliably delivering that content across SWIM in a secure fashion. It provides quality of service (QoS) including the ability to specify special handling based on priority and response time requirements. It includes functions supporting synchronous and asynchronous information exchange. Messaging functionality applies primarily at SWIM run-time.

2.1.3 Enterprise Service Management (ESM)

ESM provides management of the SWIM exposed services provided by NAS systems, as well as the management of supporting SWIM Core Service themselves. ESM includes the monitoring and control of faults, configuration, accounting, performance and security. ESM services include both design-time and run-time aspects of SWIM. Segment 1 ESM functionality is at the discretion of the SIPs .

2.1.4 Security

Security provides functions that are used by NAS systems and Core Service to ensure that SWIM services are provided in accordance with established security policies. Service security functions are used to enforce security policies and include authentication, authorization, and access control functions. Security services are necessary during both design-time and run-time for SWIM.

Security controls are selected in the process of developing an FAA ATO Security Certification and Authorization Package (SCAP). By federal policy, NIST FIPS 200 is the overall “parent” for SWIM security requirements. NIST Special Publication 800-53 provides lower- level security requirements and NIST Special Publication 800-95 contains web services guidelines for security. Also, SWIM Segment 1 Web Services security interoperability requirements are specified by the Web Services Interoperability Organization (WS-I) Basic Security Profile.

2.2 Reference architecture from SOA Best Practices White Paper

A SOA Reference Model, Figure 1 below, was described in the SOA Best Practices White Paper[1] and serves as the starting point for a more detailed examination of related Reference Models. As per the Federal Enterprise Architecture (FEA) reference model, the Service Model is reflected as a mechanism for rationalizing IT service assets within a Federal agency, or across agencies. As we will describe below in discussing shared services, reference architectures provide a view of relationships between and among functional levels within an enterprise.

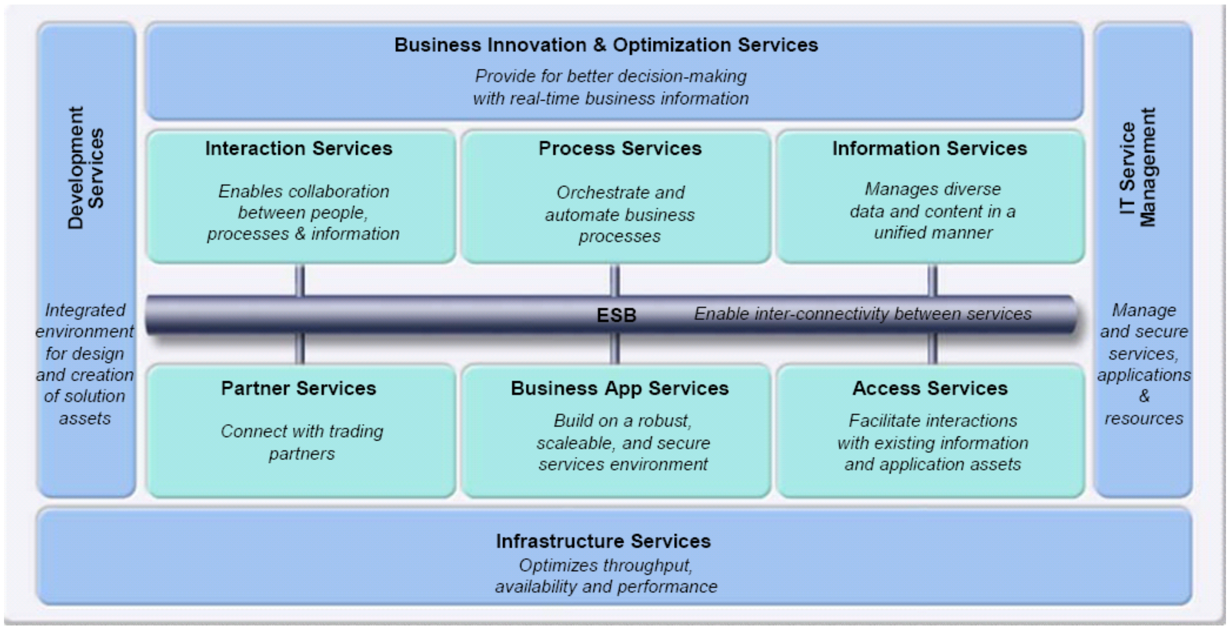

Business Innovation & Optimization Services
Provide for better decision-making with real-time business information
Development Services
Integrated environment for design and creation of solution assets
Interaction Services
Enables collaboration between people, processes & information
Process Services
Orchestrate and automate business processes
Information Services
Manages diverse data and content in a unified manner
IT Service Management
Manage and secure services, applications & resources
ESB
Enable inter-connectivity between services
Partner Services
Connect with trading partners
Business App Services
Build on a robust, scaleable, and secure services environment
Access Services
Facilitate interactions with existing information and application assets
Infrastructure Services
Optimizes throughput, availability and performance

Figure 1: SOA Reference Model from the SOA Best Practices Whitepaper

In Segment 2 we are seeking to provide the first instance of a shared service infrastructure, including Core Service. These Core Services, and the ability to share business services with members of a community of interest, will be managed centrally, but distributed across the SWIM Implementing Programs (SIPs). As currently envisioned, loosely coupling must be ensured for each SIP that will make services available to the members of the COI. In addition, strict version control, interoperability testing, load and usability testing, as part of an overall governance regime, must be enforced to clearly delineate SIP business services.

The reference architecture from the SOA Best Practices whitepaper classified services into two categories: inner and outer services. In this section we will take a more concrete view of service interaction and lay the foundation for a detailed discussion for what we see as serious problems and major impediments to program success.

2.2.1 <u>Service Layers</u>

A core concept in SOA is the notion of abstraction, or layers. From an architectural perspective, abstraction layers allow the organization to create the opportunity for reuse, composition, loose coupling and all the other aspects of large scale, systems of systems, common in large federal agencies, like the Federal Aviation Administration, the Department of Defense, the U.S. Intelligence Community, and many others. Figure 2 below illustrates how an Intra-SIP SOA reference architecture might look.

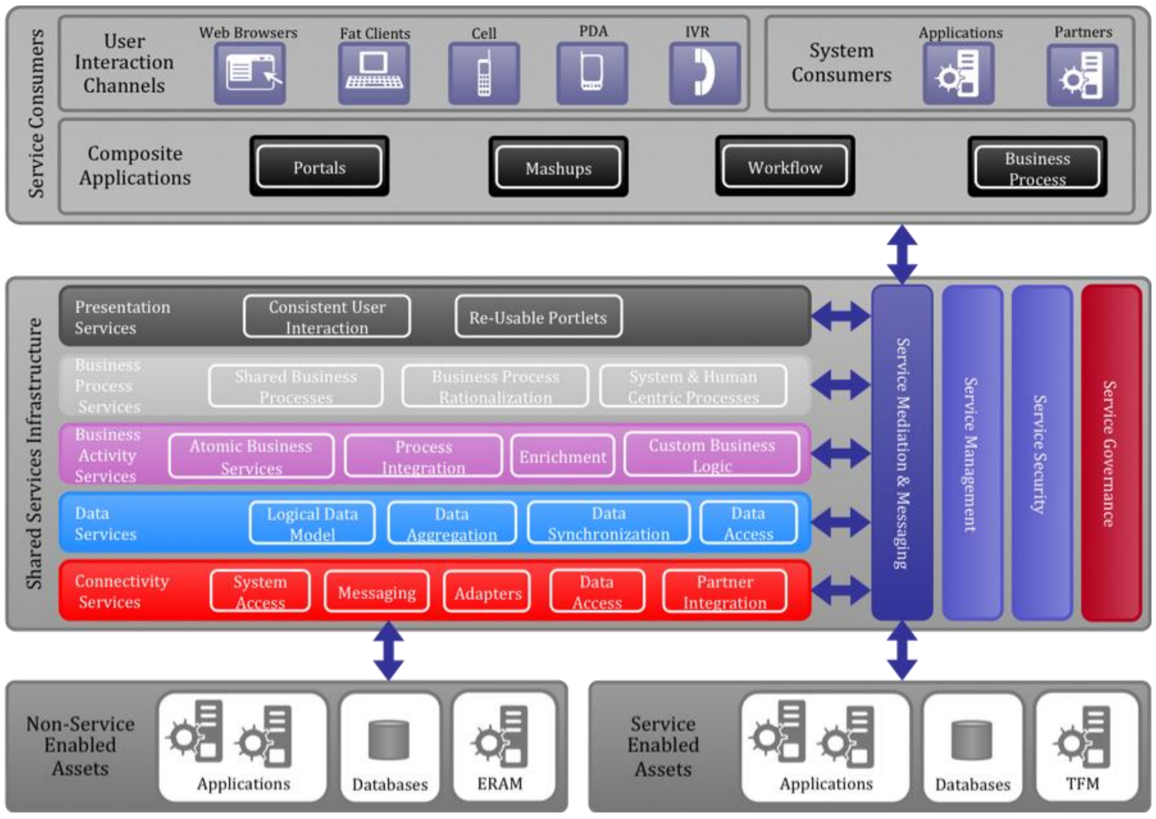


Figure 2: Sample Intra-SIP SOA reference architecture

Each logical layer abstracts functionality through a collection of defined interfaces, avoiding tight coupling, wherever possible, and enforcing standards-based interaction.

The Shared Services Infrastructure Layer, illustrated in Figure 3, envisioned for central SWIM implementation is a subset of the above diagram:

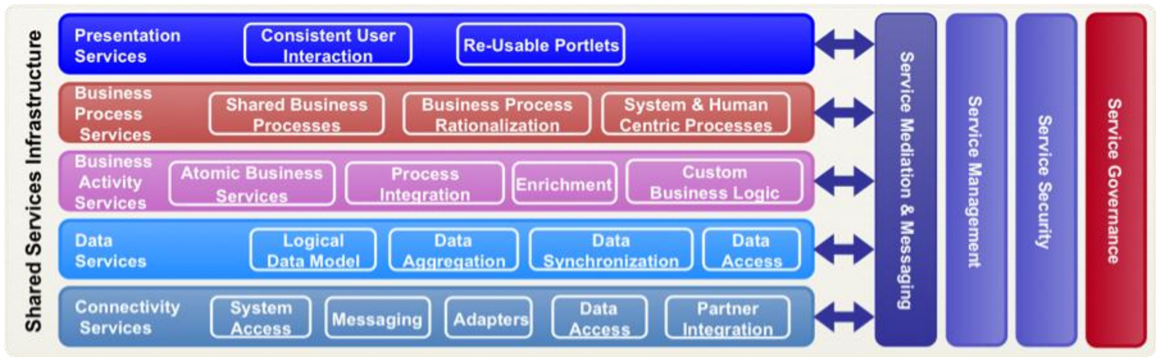


Figure 3: Shared Services Infrastructure Layers

In turn, each SIP will have to interact with other SIPs, according to a Community of Interest model that will not do away with individual program functional responsibilities, but instead promote federation of functionality and capability across each domain. Many of the Core Services have enterprise-wide characteristics and, as such, are common in terms of requirements and implementation. These common enterprise services lend themselves to sharing infrastructure to optimize NAS investment and minimize risk (both schedule and compatibility). The shared services infrastructure of an individual SIP will, in time, come to closely resemble the shared services infrastructure of other SIPs. This does not necessarily imply long-term portability of solutions, but should imply long-term interoperability. Attention to the long-term interoperability should be a key program concern.

NAS-wide services are simply another domain in the reference architecture described in Figure 4 below. Note that enterprise services can be made available directly to individual systems, as long as an intra-SIP interaction is not required. There is a notable lack of point-to-point integration in this model.

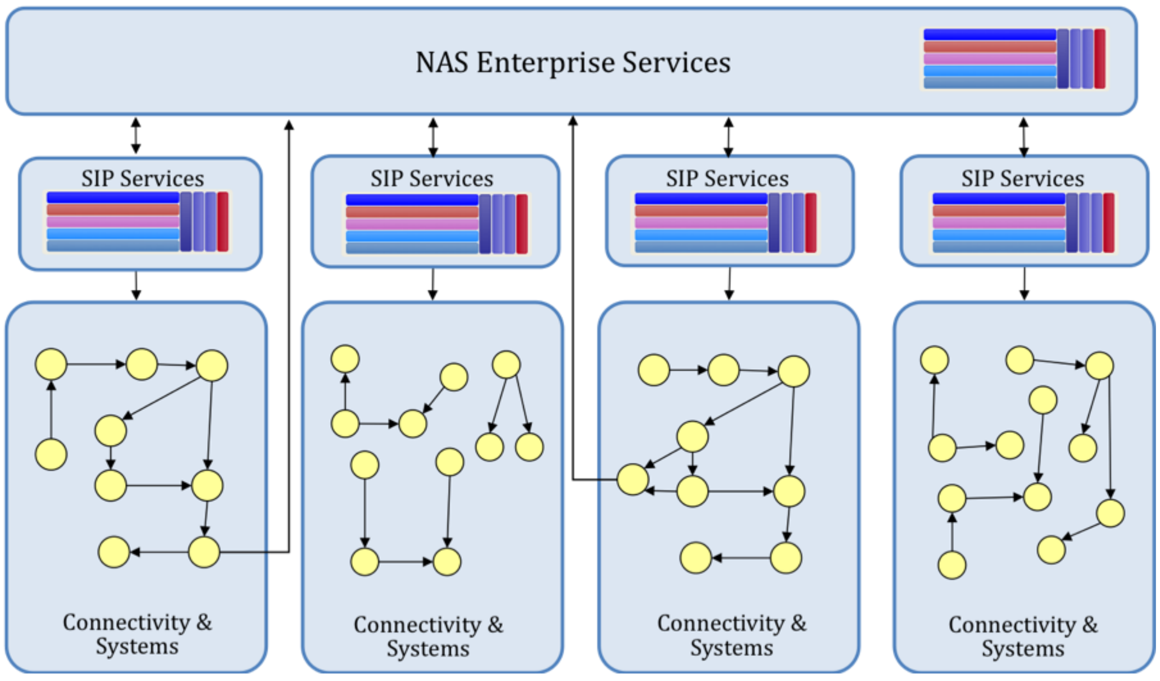


Figure 4: NAS-wide Service Reference Architecture

The SIP Services above, managed and controlled by each program, can be locally implemented, or part of a shared infrastructure model, which is described later. There are issues and challenges with each approach, as we will describe in detail.

In addition, there is a scale of abstraction to services, whether they are centrally managed or highly federated. This is depicted in Figure 5 below.

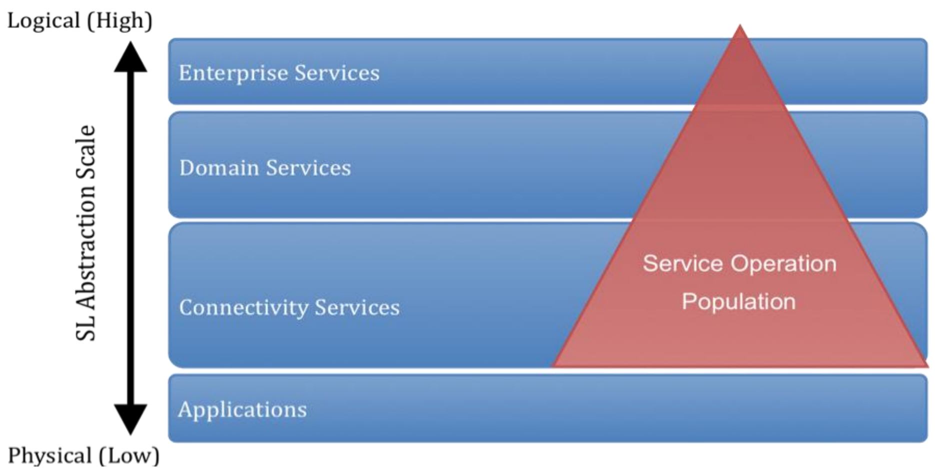


Figure 5: Abstraction of Services

Among the services that a virtual organization of the size and scope of the NAS, considering the amount of legacy applications and their importance, the largest number of services will be associated with connectivity services. Running large numbers of connectivity services entails careful attention to system interaction, by necessity. Management of connectivity services often entails substantial knowledge of network infrastructure, system-level application programming interfaces, legacy application behavior, and many other disciplines that are far removed from business-related functions. They also require substantially different composition expertise.

Tight coupling is a major problem in information exchange, and Service Bus technologies do not inherently prevent tight coupling.

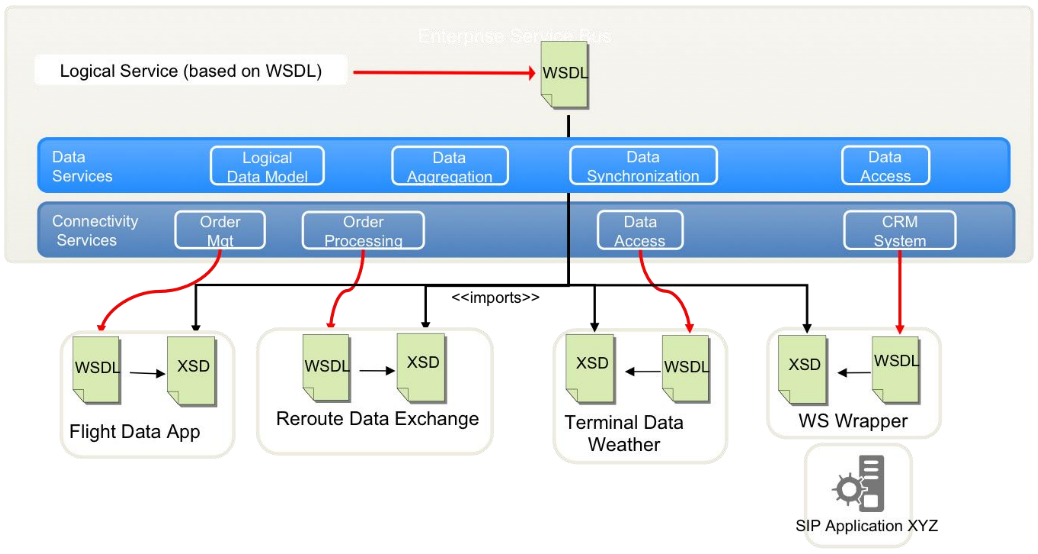


Figure 6: Enterprise Service Virtualization Advances Agile Information Exchange

For example, each time a WSDL associated with a federated service changes, the logical service must be recompiled and results in tightly coupled services. There are service bus technologies that do allow for the physical portion of the WSDL to change and that will get picked up dynamically without recompilation. The service bus should have runtime intelligence qualities. Instead, context is preserved at each level of the service infrastructure. Federated and central enterprise services can be managed discretely, and business functions and versioning coordinated across COI participants.

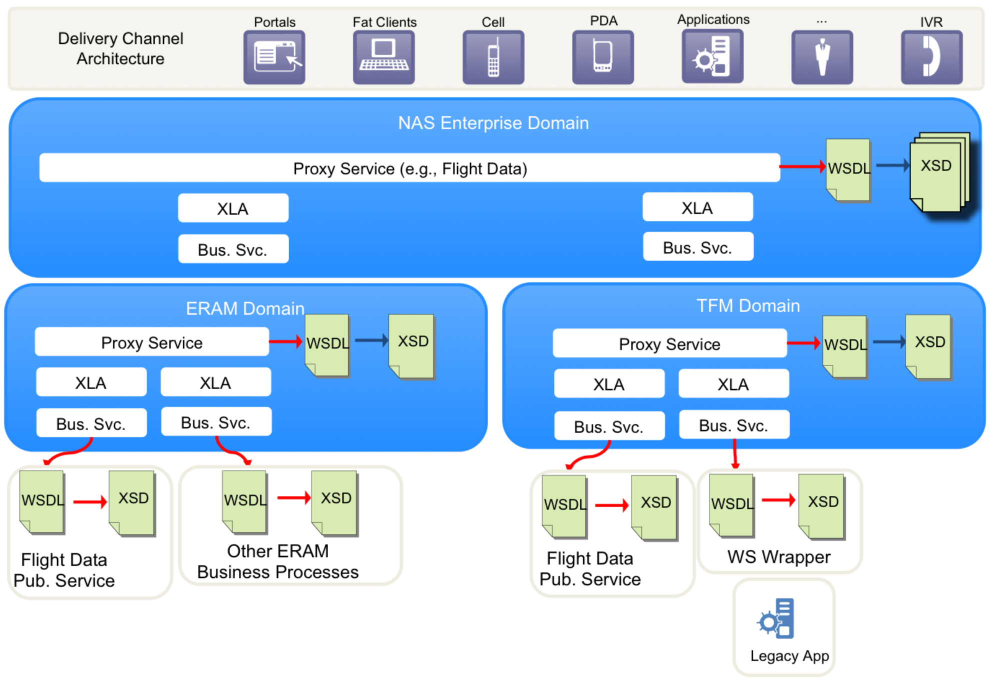


Figure 7: Enterprise Services Architecture Applied to the NAS

2.3 Discussion of What Segment 2 Should Be (from a SOA perspective)

2.3.1 <u>Current SWIM Segment One Systems</u>

This section contains a list of systems currently under consideration for incorporation into SWIM Segment One. Each of these systems will offer a service via SWIM, as appropriate, i.e. through messaging, web services, etc. In order to offer these services, the supporting SWIM Implementing Programs (SIPs) will need to leverage what are called “Core Services”. These services are common to the delivery of all data across SWIM, and include interface management, system security, enterprise service management and enterprise messaging.

Current systems are organized into nine “capabilities” as per the following chart:

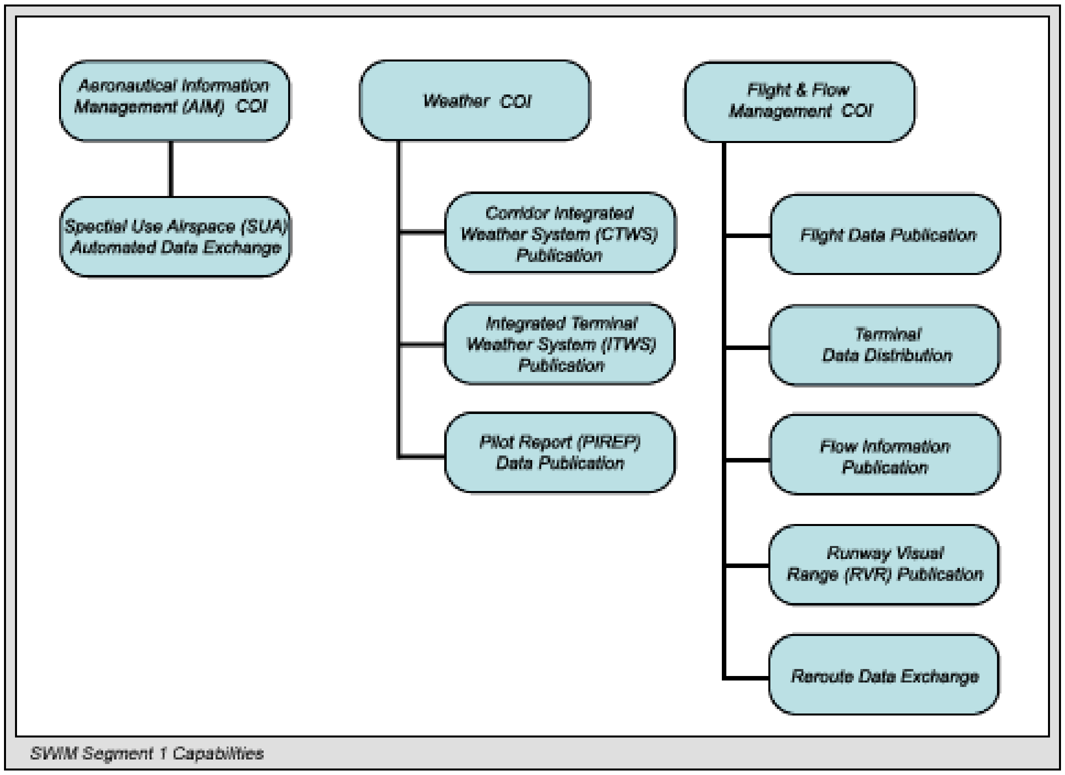

Aeronautical Information Management (AIM) COI
Special Use Airspace (SUA) Automated Data Exchange
Weather COI
Corridor Integrated Weather System (CTWS) Publication
Integrated Terminal Weather System (ITWS) Publication
Pilot Report (PIREP) Data Publication
Flight & Flow Management COI
Flight Data Publication
Terminal Data Distribution
Flow Information Publication
Runway Visual Range (RVR) Publication
Reroute Data Exchange
SWIM Segment 1 Capabilities

Figure 8: SWIM Segment 1 Communities of Interest and Supporting Functions

Each of the capabilities are provided by a system operated by a SIP (in some cases, multiple capabilities are served by a single system, resulting in seven systems total). These SIPs are:

- CIWS – Corridor Integrated Weather System
- SUA – Special Use Airspace
- ITWS – Integrated Terminal Weather System
- WMSCR – Weather Message Switching Center Replacement
    - PIREP – Pilot Report data publication
- ERAM – En-Route Automation Modernization
    - Flight Data Publication
- TFM – Traffic Flow Management
    - Flight Data Update
    - Flow Information Publication
    - Runway Visual Range

    - Reroute Data Exchange
- Terminal
    - Terminal Data Distribution System

2.3.2 SWIM Architectural Alternatives

There are several architectural approaches to building SWIM, bounded by two extremes. The first is a fully federated set of Core Services. A federated model is based upon a set of loosely coupled, self contained, individually managed entities, capable of exchanging data via interacting services by following standard protocols and governance. This implies that each implementing program builds - and is responsible for - their own set of Core Services and is consequently responsible for making them interoperate with all other Core Service implementations throughout SWIM and the NAS. This architecture is, by necessity, also distributed in nature, since each Core Service implementation is physically separate. This architectural pattern is generally exemplified by the public Internet. The Internet is built of dozens of Core Services offered by organizations who essentially volunteer to bring some semblance of order to a chaotic environment. Federation is possible through the use of standards, to which all must adhere, or cease to function on SWIM. Note that this does not necessarily imply standardized software or hardware platforms (clearly the Internet is composed of a broad range of component software), merely that the software used adhere to published standards of communication.

At the opposite architectural "extreme" is the purely consolidated approach. In this case, all core functions are consolidated and managed by a single entity. This pattern is used by many small to medium-sized organizations that have complete control over their business processes and infrastructure. Consolidation is more difficult to accomplish as an organization increases in size, since there is necessarily more use of heterogeneous systems to solve application problems. It is important to note that the use of standards helps ensure communications applies equally to both approaches. Standards are used in a consolidated architecture; they are merely used internally to the managing organization, and are not necessarily published or open for comment. Additionally, consolidation does not necessarily imply centralization, as many consolidated infrastructures are distributed in nature due to physical redundancy requirements and to manage communications costs.

There is, of course, a third architectural approach: that of "shared services". Using shared services, each core service is evaluated independently and a decision made as to the required degree of consolidation versus federation. In some cases, it may make a great deal of sense to consolidate a service and have it managed by a single entity; in other instances, a more federated approach may be desirable. For example, specialized services such as a rules engine, high-performance compute grid or rendering engine may be intensively used by one SIP and only occasionally by one or two others. In this case it makes more sense to have a single SIP provide the service for all in a federated manner. In contrast, a service such as a registry should probably be set up by a central authority and be used by all SIPs. Note the intentional use of the word "more" in describing federated. Ultimately, the shared services approach is the gray area in the middle of the two extremes. Each service may be placed somewhere in that gray area, according to functional requirements.

2.3.3 <u>Relevant Issues Associated with Alternatives:</u>

The key variables that come into play when discussing the alternatives of Federated versus Consolidated are diverse: cost, schedule, and technical all come into play.

- <u>Cost</u>: An obvious difference between the two alternatives is cost. If a federated approach is taken to implementing services, the necessary equipment required would be increased (as each federated site would require a set). At a minimum, a suite of equipment and software that would comprise the Enterprise Service Bus would need to be purchased, fielded, operated and maintained. Although a consolidated approach would reduce the amount of software and equipment required, it would not be a linear reduction as a consolidated location would require the necessary "horsepower" and requisite backup equipment to service multiple facilities. One needs look at any modern datacenter to recognize that consolidation does not necessary imply inexpensive technical solutions.

- <u>Schedule</u>: With the increased cost of equipment required in a federated approach comes the related cost in time – deployment teams are typically small, tactical teams of highly trained individuals that are sent to multiple locations in serial to plan and execute the deployment of new equipment. Because of the serial nature of their work, schedules for deployment can be stretched by months to accommodate all sites coming online. Although this isn't necessarily a bad thing, it does increase the complexity of the enterprise that is required to operate current and back versions of software and hardware interfaces depending on where in the waterfall a site finds itself. However, it must also be stated that a more sophisticated, parallel approach may be undertaken by more than one team in order to stand up service infrastructure in multiple SIPs, simultaneously. A consolidated approach would minimize the disruption, by reducing the number of sites to visit for deployment, but would likely add complexity by making the consolidated site very large and (by reason of numbers) critical to success. Depending on the final architecture of the enterprise, it is likely there will be at least 2 consolidated systems providing services to the NAS.

- <u>Technical</u>: For the FAA's purpose, it can be reasonably argued that the technical aspects concerning the deployment of federated versus consolidated are the most unknown, and therefore highest risk. Employing the use of services across a Wide Area Network that sees multi-cyclical variances (daily, weekly, monthly, and annually) in the amount of traffic (both air traffic and bandwidth) is an unknown that must be explored through modeling, prototyping, experimentation and analysis. Additionally, challenges in implementation policies, configuration management, security and the crafting and tracking of Service Level Agreements all need to be explored and incorporated into the design. It is well understood that a local version of service will provide better performance in terms of computational timeline analysis, but it is not known if that performance will meet the strict the timeline requirements for operational data. Architectural decisions regarding multiple legacy applications will be driven without verified empirical data to validate the decisions and could lead to technical dead-ends (those that will not work at all), or worse, a change in design that works but does not meet requirements and is difficult to undo. These aspects of the introduction of SOA services into the NAS must be addressed in a definitive manner through extensive modeling, prototyping, and experimental analysis. Further, the results must be vetted with decision makers from the technical, air traffic, and executive management organizations in order to maintain the integrity of the NAS, SWIM strategic direction, and the development of NextGen as a viable follow on architecture.

It is recommended that key studies be launched that provide framing data to the problem space of SWIM through federated and consolidated services – it is only through facts and analysis that FAA will be able to ask appropriate questions in order to reach useful and pragmatic solutions favored by the hybrid, shared services model.

2.3.4 Selection Criteria and Justification

The criteria for selecting an architecture for SWIM are many and varied. The selection criteria must allow for a fair comparison of the various approaches in terms of both strengths and weaknesses. While they do not capture the totality of the decision-making process, the criteria listed below offer a high-level summation of the most important decision factors.

- Flexibility – Does the architecture allow choices for the SIP? Are the standards upon which it is based open and available to all?
- Service Portfolio Management – How effectively can services be governed for the enterprise such that they are readily discoverable and usable to promote reuse or/and orchestration of services to provide more advanced capabilities?
- Enterprise Management and Service Desk Support – How effectively can enterprise services be monitored at run-time to ensure SLAs are being met. How effectively can problems be detected and root causes be determined when Enterprise Services or supporting infrastructure components fail?
- NAS Asset Utilization – How effectively are NAS processing elements providing SWIM Core Services utilized?
- Policy Management – How well does the architecture facilitate effective policy management to enable effective enterprise-wide governance?
- Scalable – Is the architecture easily scalable beyond Segment 2? Is the architecture capable of expanding to grow in conjunction with the anticipated the NAS, as data exchange will increase at increasing rates? Will it be able to support new Operating Systems, new architected SIPs, new standards, etc.?
- Efficiency – Does it optimize IT assets and network bandwidth?
- Proliferation of data – Ease of new users accessing and developing new services.
- Secure – Is the architecture easy to secure from end-to-end? Are there service level and data security level capabilities?

- Performance and Availability– Does the architecture lend itself to good system-wide performance, with few bottlenecks? Will the architecture support highly available services?
- Maintainability – Does the architecture lend itself to cost effective maintenance?
- Reliability – Does the redundancy model support reliability requirements?
- Initial Cost – How much of an investment will be required to implement the architecture?
- Management Cost – How much ongoing investment in terms of both time and material will be required to keep the architecture running?

These criteria are a generally accepted list of architectural considerations that must be taken into account when designing distributed systems, in general. The list is based on decades of industry experience yet, must be uniquely stack-ranked based on an agency's or an enterprise's specific needs. We therefore recommend that a trade study be conducted to determine the exact order of importance of each of these criteria to SWIM segment 2.

Once the quantified determination of each criterion has been made, that ordered list will become the core of an architectural trade-off matrix (a key governance tool) that can be referred to for any contentious design decisions. Each of these criteria have been listed and evaluated for each approach in the next section.

2.3.5 <u>Recommendation</u>

Based on the initial evaluation and grading of selection criteria in section 2.3.4 of this document, the Shared Core Services architectural approach appears to be more beneficial to the FAA for SWIM. Figure 9 below illustrates a graded comparison of the 3 architectural models for each of the 14 selection criteria.

	Federated Core Services	Consolidated Core Services	Shared Core Services
Flexible	3.6	2.9	4.7
Initial Cost	2.1	3.8	4.1
Management Cost	2.9	3.9	4.2
Service Portfolio Mgmt	2.7	3.8	4.1
Enterprise Mgmt & Service desk	2.4	4.4	4.6
NAS Asset Utilization	2.8	2.8	4.8
Policy Mgmt	2.3	4.1	4.4
Scalability	3.3	3.3	4.9
Efficiency	2.9	3.9	4.6
Data Proliferation	2.7	3.9	4.6
Secure	3.3	3.8	4.4
Performance	3.4	2.9	4.6
Availability	2.8	3.7	4.3
Maintainability	2.2	3.4	4.0
Overall Grade	39.4	50.6	62.2

Grading

- 5 – Offers outstanding benefits
- 4 – Offers better-than-average benefits
- 3 – Offers average benefits
- 2 – Offers below-average benefits
- 1 – Fails to offer any net benefits

Figure 9: Comparison of Service Models for Segment 2 and Beyond

This scored evaluation was obtained by asking all of the participants in this working group, identified in the table of contributors at the top of this document, to submit their evaluation using the identified criteria. All of the criteria were evenly weighed for the purpose of this evaluation. The objective is to select a good architectural paradigm for Segment 2. The scores for all respondents were then totaled for each criteria and each model and then divided by the number of respondents, thus the mean of the responses is what is presented in Figure 9.

Shared Services seems to buffer some of the risks associated with the extreme approaches, while at the same time not imposing significant risk itself.

Shared Core Services offers more flexibility than the purely consolidated approach, by allowing the SIP to choose which services they implement themselves and how they implement those services. The SIPS are allowed an even greater degree of freedom with a federated architecture, since they are not constrained at all; of course, this shifts a greater burden to the program itself as this potentially incompatible array of software must be coaxed into communications. This problem is ameliorated somewhat (but not completely) by the use of standards for SWIM, which is the Shared Services approach – offer the greatest amount of flexibility to the SIPs, while requiring an adherence to standards that facilitate communications. In reality, deviation from the standards on the part of any SIP will result in that SIPs' services failure to be consumed by other SIPs.

Initial cost is fairly incomparable with all three approaches, with the investment being made in different areas according to architecture. The consolidated investment is focused in building software at the core, to be managed by the SWIM program. The federated investment consists of building bridges between redundant systems and federators for the SIP services, managed by the SIPs. Shared services is a combination of these two approaches, with some services consolidated and managed by FAA, and some federated and managed by the SIPs. The cost, associated with standing up each architecture, is directly related to the ability of each architecture to re-use existing NAS and SIP resources.

Management costs are extensions of the above logic: anything that the SWIM program needs to manage increases costs. The key differentiator is "What will be managed?" The shared approach allows services that make sense to be implemented and managed in a unified fashion, while letting the SIPs manage the remainder. Federated management cost will be substantially higher than a shared services model since redundant systems will also require redundant staff plus, the bridging and federating must be managed and maintained. Consolidated could possibly be less expensive in the short term, however, there will be limited benefits from the natural efficiencies that the specialization of a shared services model enjoys and will therefore be more expensive over the long term. Finally, the FAA has already made a number of decisions with regards to organizational structure that will make a shared services model somewhat more familiar thereby decreasing the need for broad change management.

Service portfolio management will be equally challenging in both consolidated or shared services architecture. Nevertheless, it will be more of a cumbersome task in a federated architecture due to the replicated provisioning of services. A large aspect of portfolio management is human communications and in a federated environment, every participant is required to know the same information at the same depth/breadth at the same time. The aspect that tips in the favor of shared-services is that each SIP has self-interest in managing their own portfolio as effectively as possible.

Enterprise Management & Help Desk really shows no clear winner between the consolidated and shared-services architectural models. The difference primarily comes down to the philosophy of operations: While the overall manageability and operation of a physically and logically consolidated system may be obvious, the modularity and physical separation of highly componentized, shared services is much simpler to troubleshoot and repair. Reactive management practices favor the former whereas preventative/predictive management practices favor the latter.

NAS Asset Utilization in the shared-services model is the most pragmatic and utilitarian by a wide margin. This is primarily because the existing infrastructure can be used as a reasonable baseline and there does not need to be a large, initial build-up of infrastructure to meet some minimum set of requirements. Existing infrastructure can be exploited through re-purposing and re-provisioning.

Policy Management is similar to portfolio management in that the degree of challenge is approximately equal in both the consolidated and shared-services models, but substantially more challenging in the federated model. Essentially, the same reasons apply.

Scalability favors shared services by a substantial margin. A federated architecture shifts the n-squared problem from the application to the federator; we will still need to build bridges between applications. A consolidated architecture will require a constant upgrade stream to maintain functionality through several SWIM segments. Shared services make the distribution of services much more discrete, thereby bounding variability. Unknown variability is one of the major hurdles in predictable scalability. This means that additional resources only need to be applied to those distinct services that need it instead of wholesale, system-wide upgrades.

Efficiency is optimized in the shared-services model. As noted before, the federated model requires massive amounts of redundancy, regardless of SIP-specific need. The consolidated model requires a one-size-fits-all approach which means a varying degree of over and under-utilization of resources which can only be mitigated by very complex capacity/performance planning processes. As in scalability, the efficiencies of the shared-services model is that resources only need to be applied when and where they are needed with a high degree of precision.

Proliferation of Data also shows the shared-services model to be the clear winner. In the case of a federated model, most data will need to be replicated and synchronized on a regular basis. In the consolidated model, this shouldn’t be the case but, our years of experience with human behavior tells us otherwise and data will proliferate, over time. The shared-services model requires clear lines of data custodianship however, by its very nature, also requires data sharing amongst peers. Conceivably, this is the best of both worlds provided the governance mechanisms put in place are consistently monitored.

Security is both a core service and a concept. We focus here on the concept of end-to-end security from information source to information consumer. The consolidated architecture is by far the easiest way of delivering on this concept. By definition, a consolidated system has complete control over all the processes within SWIM, and can manage security accordingly. Shared services leverage that concept by consolidating and sharing a good deal of that security functionality, such as global authentication and identity control, leaving the SIP to use this information as required. Federated security is by far the least effective approach. Federated security implies that each SIP will maintain their own security infrastructure and hopefully share such information in a federated fashion. This is both ineffective and wasteful, since it does not support end-to-end transfer of data, and the vast majority of the infrastructure will be unnecessarily duplicated.

Performance is a clear winner for shared services. The shared approach allows those services that benefit from physical diversity or proximity to the user to be deployed in that fashion, whereas the extreme architectures force the location of services to one degree or another. This prevents bottlenecks like bridges and also reduces the risks associated with long-haul communications to obtain a service.

Maintainability shares many characteristics with enterprise management and scalability and thus favors the shared-services model. The combined benefits of specialization and service expertise plus clearly delineated boundaries for variability delivered by shared-services outweighs the benefits of physical proximity in the consolidated model.

While this is a very cursory examination, it seems clear that the Shared Services architecture offers a reasonable number of advantages to the other, more extreme approaches, and thus should be chosen as the model for SWIM Segment 2.

2.3.6 <u>Operations Concept (OPSCON)</u>

All of the documentation and information available on NextGen leads to two (2) conclusions:

1. There will be a high demand for information, needed ubiquitously
2. More sophisticated planning tools will drive system complexity and the scale of available data.

Each of these capabilities has consequences in information exchange and interdependency between NAS Applications and their respective communities of interest (COIs). Accordingly, it is no mistake that network-enabled information access is at the top of the Joint Planning and Development Office's (JPDO) list of eight (8) key capabilities[2].

Network-enable information access is the foundation layer that enables the data exchange that supports the FAAs distributed and more highly automated decision-making paradigm, with considerable implications to NAS operations. Transitioning to this paradigm will result in an information explosion, where available data will grow by orders of magnitude, requiring scalability and flexibility for all NAS stakeholders. This growth in data will yield technological innovations that create new information, and needed now to achieve the requisite airspace utilization and capacity breakthroughs. SWIM requires delivering timely and ubiquitous data, available in common structures and consistent formats to attain the lifecycle savings and potential system innovations needed to achieve NextGen.

Today's systems, mired in legacy workflows, are not designed to scale beyond current ATC-centric operations. They were built around a "controller-centric" architecture, whereby data is processed by systems to serve information to the NAS decision-maker. That design is currently constraining NAS expansion, limiting NAS scalability to the processing capacity of the decision-maker, hindering increased capacity, utilization and NextGen. Although looking for solutions that solve today's data exchange problems is important, it will not suffice. NextGen demands a radical approach that's flexible enough to handle immediate data exchange needs, with the scalability to solve the more highly automated workflow requirements of tomorrow.

Today, user requirements emerge based on mission needs. Requirements are collected, features evolved and NAS Applications enhanced by costly engineering change proposals (ECPs), over prolonged schedules. Architectural decisions constrained by the most contemporary development tools and practices, force users to accept capabilities based on requirement interpretation and system trade-offs. Modern development tools and practices, inclusive of service-oriented architecture (SOA), offer many of the protocol abstraction capabilities required to achieve the "common formats and standard structures" needed for an improved information exchange. Technology has evolved, now capable of abstracting data and exposing it to broader communities of interest. More flexible architectures can leverage existing systems to expose their respective datasets and migrate to a more agile development environment.

2.3.6.1 OPSCON: Segment 1 Service Container

SWIM was envisioned to provide those capabilities that facilitate, and, expedite data interchange, and accelerate system interoperability. Using a service-oriented architecture, the FAA envisioned achieving much needed operational improvements and NAS wide governance, leading to the construction of a more agile NAS. The current SWIM Segment 1 architecture deploys independent infrastructure across each SIP. An excerpt from the SWIM Technical Overview[3] describes the planned environment:

> "The plan for Segment 1 is for each SIP to design, implement, and deploy NAS System SWIM Servers to provide the interface between the systems that are the responsibility of that program and the other systems participating in SWIM Segment 1. The SWIM program will provide standardized Service Container software which the SIPs will deploy to NAS System SWIM Servers."

SWIM Segment 1 introduces a federated architecture, where each participating SIP designs, develops, deploys and manages part or all, of their "SWIM infrastructure". This infrastructure is made up of two (2) functions:

> (1) IT infrastructure function: simply, the hardware and operating system (and components) required to host the SWIM Service Container; and,
>
> (2) SWIM Services function: the suite of software applications that make up the Service Container, loaded on the IT Infrastructure and used to provide the SWIM Core Services.[4]

SIPs have the responsibility for procuring the hardware and operating system components (IT Infrastructure Function), which will be required to meet the specification of the "standardized software" package. The SWIM Services Function is made up of a suite of software components that compose a Service Container. The Service Container, recently awarded to Progress Software's IONA FUSE product suite under an IDIQ (indefinite delivery, indefinite quantity) contract vehicle, will be provided to the SIP by the SWIM Program. Each SIP is responsible for selecting the software applications that will make their Service Container SWIM compliant, and hence, capable of exchanging NAS information. In this deployment method, the SIP is responsible for their own implementation and maintenance of the IT infrastructure and SWIM Services functions. SIP responsibilities will include configuration management, service management, MOM component, life cycle support, and security for the Core Services as part of their planned release upgrades, or as part of their new acquisitions[5].

The SWIM Program involvement is in providing the SIPs the components and requisite training for implementing the Service Container. Similar to today's implementations, the SIP will select the applications, make the architectural decisions for all aspects of the Service Container. Considerations for standardization will yield to program requirements, interpretation and non-technical factors (schedule considerations, etc.) that may compromise deployment decisions. This deployment model provides over-arching control of the software suite available for the SIPS in the hands of the SWIM Program Office, and ensures "standardized" Service Container deployments across the SIPs do not become a massive configuration management task.

For SWIM Segment 2 and beyond, the architecture and deployment model in the segment 1 timeframe can lead to point-to-point limitations that SWIM is intended to solve. Without some shared service components, like a common registry, application programs will have difficulty discovering NAS-wide available products. Those application programs seeking to exploit available products will, invariably be required to interface with the program offices to establish a subscription to those desired products and services. These become "virtual point-to-point" interactions between participating application programs. This is understandable at this nascent development point of the NAS architecture and implementation of the program.

It is imperative that, eventually, some functions are migrated to a more logically centralized management model: governance, enterprise service management, system security and enterprise messaging can be leveraged across multiple users to apply consistent, standardized policies across the SIPs. As such, these capabilities provide a shared utility service common across all NAS Applications. Approaching the standardization of policies must be done from an enterprise perspective, to protect against any potential silo-building. As this model matures it is important that the mechanism that specifies changes (the requirements process) be updated to take full advantage of the centralization of the common functions to yield a more enterprise-consistent methodology, versus managing an Enterprise-centric structure with application-centric engineering.

Consider the following example: Service Container Implementation for two terminal areas. Each terminal has a set of applications that run on a program-specific bus (PSB) (Enterprise Service Bus, or Bus) to deliver information sharing capabilities utilizing SOA services within a specific program, or targeted at a particular Community of Interest (COI), like Weather, or Surveillance, etc. To facilitate operations, data is passed between one terminal system in a PSB (program-specific bus) utilizing some SOA environment. The second terminal has a similar deployment utilizing another SOA PSB for intra-facility information exchange. Despite the fact that both these programs have their own instantiation of a SOA for their respective implementations, through the Segment 1 Service Container, they will deploy another messaging capability or SWIM-standardize the existing messaging capability to interact with each other via the Service Container. The following graphic illustrates the example listed above, where each SIP utilizes publish/subscribe, or request/response messaging methodologies, via MOM and Web Services technologies, in order to exchange information via their respective Service Containers.

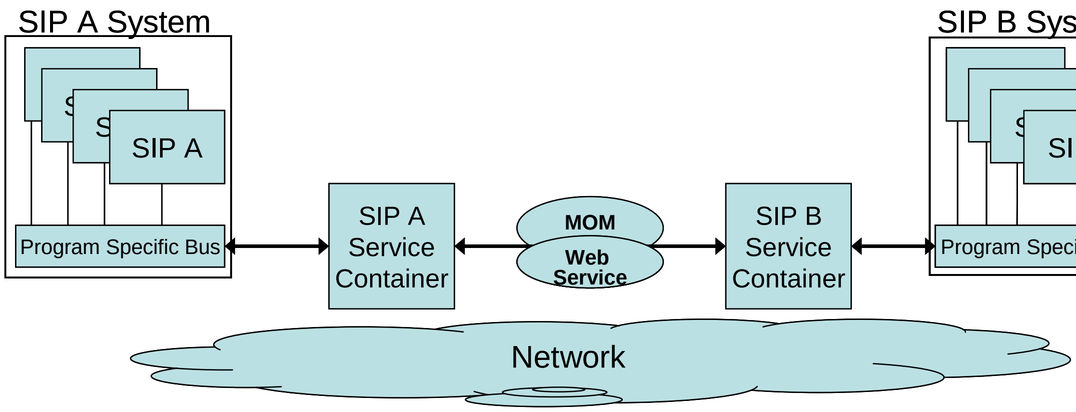


Figure 10: SIP self-provision of messaging methodologies

In this model, the first terminal system is made up of several boxes, representing the geographically dispersed nature of the installation across NAS facilities. The PSB provides the messaging exchange between systems. The second terminal is similarly represented by a series of systems distributed across the NAS, exchanging information across another SOA PSB. All of these sites and systems are tied together by the network. Both sites will be required to use a Service Container, each selecting the series of applications needed to achieve the SWIM requirements. Each system will have to configure their respective PSBs to interact with their individual Service Containers. Once each SIP is publishing, or exposing services to the Service Containers, both SIPs may elect to publish and subscribe to each others data. Depending on the SIPs chosen implementation, for example, via a JMS requiring a message oriented middleware (MOM) (described in detail in Section 3), or by exposing some data element via a web service, each will require varying levels of collaboration to achieve the integration benefits offered by SOA.

A potential issue arises because the selection of a MOM product for transmission of messages among each SIPs Service Container is left up to the SIPs. If each SIP selects a different MOM product then there will be interoperability issues among the SIPS or at the least potential duplication of capabilities in order to interoperate. Even if SIPS select the same MOM product, there may be revision level compatibility issues that must be carefully coordinated among SIPs.

The configuration management issue is further complicated by the disparate applications that make up each individual SIPs Service Container. As each SIP interprets the SWIM requirements and manages their implementation, there is risk with maintaining revision control across the various applications that make up the Service Container. Each SIP must be committed to maintaining this peripheral data exchange environments, and coordinating with all other SIPs for planning upgrades, while maintaining their individual environments.

This leads us to the issue of program asset utilization. Each SIP is charged with an FAA mission critical capability. Despite the fact that weather, for example, is classified as an "essential service", it is critical to the success of maintaining an operationally efficient NAS. These SIPs have a dedicated staff (NAS resources/assets) to support the basic functions of these mission critical systems. TFM, for example, currently charged with maintaining traffic flow information for the Terminal environment, has built its own Oracle-based PSB. These resources will now be burdened with the additional responsibility for building out and maintaining the Service Container environment, a different suite of software applications providing similar capabilities as their PSB; however, using a different vendor implementation. These additional requirements detract resources away from the SIPs primary mission: managing traffic flow in the case of TFM, for example.

Commercial implementations rarely diverge from this type of federated administration. Infrastructure that supports enterprise-wide needs is typically classified support systems and deployed by an information technology group. Each location is also required to have on staff the necessary personnel to administer the local differences between the corporate and business unit software package. This approach centralizes some administrative functions while keeping specific capabilities needed by local personnel directly in their control, creating a combined federated and consolidate approach. This type of mixed administration approach is required in the commercial environment, because commercial enterprises are held accountable for Return on Investment (ROI) and Return on Assets (ROA) metrics, demanding streamlined yet nimble operations and effective utilization of available processing capacity.

Enterprise Service Management supervises the infrastructure providing the monitoring and control function of the IT Infrastructure and SWIM Services functions, within the SOA environment. In the event of a message exchange failure between two facilities, the two sites will have to work together to identify and resolve the root cause failure. By using a consolidated ESM, this fault analysis can be done centrally and increase system repair time.

These points draw us to one conclusion: there are some functions that lend themselves to a centralized administrative strategy, where common functions are shared amongst all users. This is the basis of the working group's recommendation for Shared Services (with some service provided by the SIPS directly, and some provided through the WAN by consolidated services). The following section provides a concept of operations for transition to a Shared Services Model, the benefits of which include spreading administrative and licensing costs across multiple users, maximizing IT infrastructure utilization, integrate monitoring and control functionality, facilitating NAS-wide information sharing governance, while empowering SIPs (operational departments) to stay focused on their mission critical activities, maximizing program resources, among others.

2.3.6.2 Segment 2 OPSCON: Shared Service Model

Team discussion was supported by NextGen and SWIM concept documents, which emphasize the need for a collaborative air traffic management system (C-ATM), featuring a high degree of information sharing via net-centric capabilities. The team equated NAS Applications to operational departments within an enterprise. In that context, shared infrastructure is a common strategy to maximize organizational investment and leverage re-use. Moreover, one of the key advantages to SOA is re-use, which lends itself to a net-centric capability, consistent with the FAAs vision. As a result, the group spent several conference calls discussing the various architectural strategies to determine which would create the greatest value benefit to the FAA. It is through these many hours of discussion, utilizing the criteria characterized above, that the team arrived at the Shared Services Model.

This concept centralizes some of those functions that are commonly used by all participating NAS Applications.

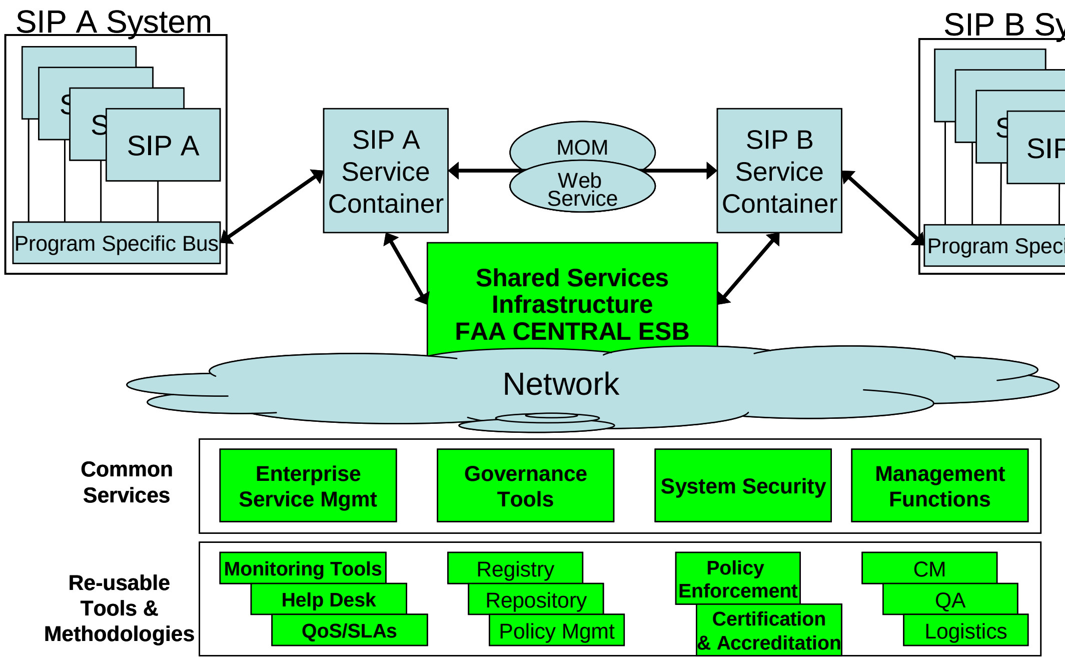


Figure 11: Shared Services Model

The FAA Central ESB provides an integrated common messaging platform that can be used and accessed by any NAS Application for exchanging information. As SIPs engage to exchange information via the Central ESB, they will conform to a standardized "on-ramping" process. Once on-ramped, these users can expose parts of their infrastructure, an algorithm (as a Web Service) for example, to other SIPs, and become empowered to find ways of leveraging underutilized assets. Additionally, providing the Registry and Repository, tools and process to manage policy development and enforcement, ensures a consistent mechanism for governing enterprise-wide data exchange. Centralizing the Enterprise Service Management (ESM) enables the establishment of SLAs for enterprise-wide services that can be effectively monitored and enforced resulting in accountability for service performance delivered by service providers. A Centralized ESM eliminates many monitoring gaps within the multiple failure points within the information sharing environment. Employing tools that bring all those components under single management prevents the typical "finger-pointing", identifies and corrects root cause quickly. A centralized ESM enables the FAA to more quickly identify failures within a messaging chain, accurately source root cause and establish SLAs to hold those accountable who miss their obligations. System security functions are also important. In a Central ESB shared services model, SIPs would define the policies in accordance with their mission and security/ESM systems enforce those policies.

Meanwhile, the SIPS are free to establish the Enterprise Service Bus that meets their needs to deliver highly critical NAS operations data across the WAN with efficiency and agility. By implementing their own ESB's for things like messaging (if necessary) and applications services the performance of the NAS is enhanced.

Additionally, the management functions of this type of capability are often overlooked. The decentralization of the Service Container administration across many small facilities could lead to potential compatibility issues. Ensuring that Service Containers are tested, verified and qualified against some standard is important and the management of that function can be centralized for those facilities lacking adequate IT personnel, skills, or equipment. Maintaining the IT Infrastructure and SWIM Services functions are left to the SIP, whose responsibilities will be tied to ensuring their mission applications are running first, any data sharing second. SLAs can be drawn, based on some quality of service (QoS) constructs agreed to between SIPs; however, without a centralized ESM, tracking performance against those SLAs is a non-trivial matter.

These functions are important to the successful administration of the NAS, and lend themselves to be shared across multiple SIPs. The purpose for developing this shared services model, is to abstract the infrastructure, underlying protocols and system-specific details, providing an open, standards-based SOA that enables drastically increased information exchange among NAS Applications in the form of well structured services. An industry example of this is the Service Component Architecture's concept of a Data Service Object. A Data Service Object is essentially a representation of any query-able data source (database, file system, etc.) made available as a web service. At its simplest level for example, a stored procedure within DB2 can be exposed as a web service which is deployed as an end-point on the bus. This enables the composite application developers to be free of hardware/software-specific details of data access.

The shared services model abstracts the underlying IT Infrastructure and SWIM Service Functions integrating these elements into the network and exposing them to an end user as a service. This is the foundation of net-centric operations and will yield additional operational efficiencies and opportunities for re-use. The shared services architecture leverages embedded data distribution native to the COTS technologies readily available in the marketplace. Additionally, tools for those critical functions that lend themselves to a more centralized management model can be achieved. Governance, enterprise service management, system and security and enterprise messaging can be leveraged across multiple users to apply consistent, standardized policies across the many NAS Applications.

2.3.6.3 Segment 1-to-Segment 2 Transition OPSCON

The working group determined it prudent to propose the deployment of a consolidated services model to enable the FAA to migrate toward a more centrally managed infrastructure for smaller facilities and SIPS without disruption to the operations of the SIP.

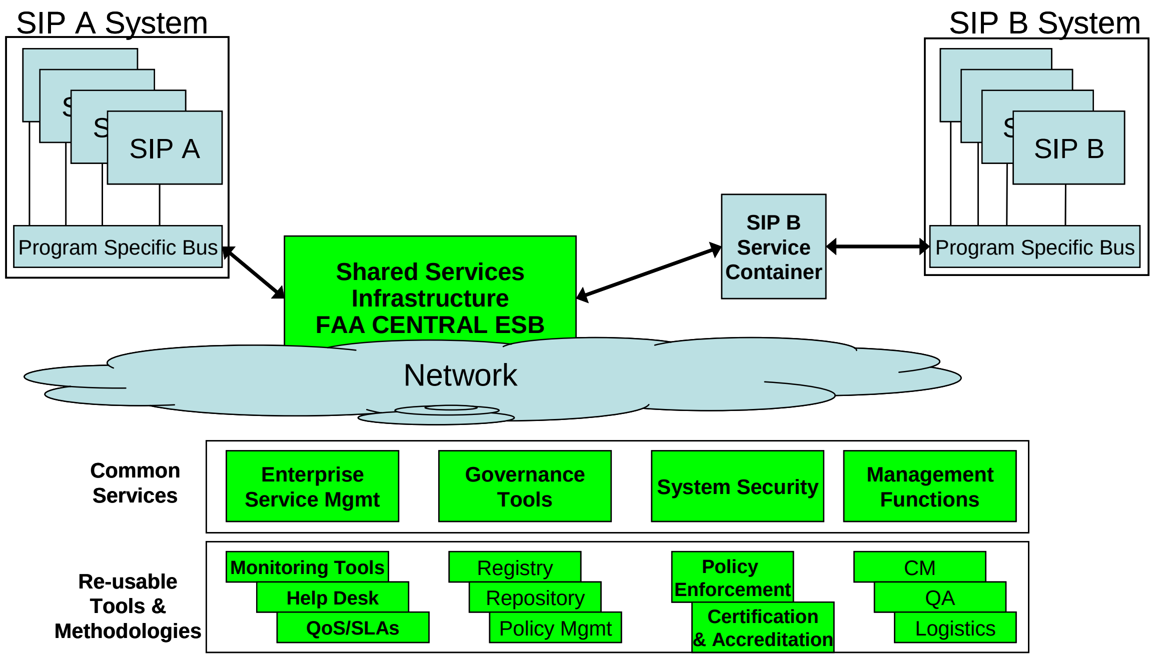


Figure 12: Transition from Segment 1 to Segment 2 Services

A graphic portrayal of the transition from a federated Segment 1 Service Container approach to an FAA Central ESB offers a logically consolidated enterprise service bus. Although physically distributed, this model standardizes the on-ramping of data and service providers, while ensuring consumers have a selectable source to acquire their desired data. By establishing a standardized Central ESB, SIPS seeking to exchange information have access to a platform that enables interoperable formats and standard structures.

Segment 2 "SIPs" (those Segment 2 SWIM Implementing Programs) that require centralized services would on-ramp to this standardized platform. Section 1.3.6.4 provides more details about those programs. However, for the Segment 1 SIPs and any Segment 2 SIPs requiring their own federated services, a transition to a Central ESB would not be necessary as they are deploying them natively. Any such SIP could decide to remove their native Service Container, choosing instead to on-ramp to the consolidated messaging and web services if it was found to be to the FAA's advantage at some point in the future. From their existing PSB they would employ a standardized mechanism for exposing web services, or enable enterprise-wide messaging. Essentially, the Central ESB would provide a single messaging interface standard that would require each SIP to implement within their environment without the responsibility of transforming to the Service Container's messaging service.

The benefits would be immediate with the FAA Central ESB, enabling the FAA to leverage the common management requirements for the architecture. Section 3.5 and 3.6 provide an in-depth characterization of this transition from Segment 1 to Segment 2.

2.3.6.4 OPSCON: Growing Information Exchange in Segment 2

The shared service bus, offers an environment where a physically distributed capability could be logically consolidated to a Central ESB, providing a NAS-wide message exchange. NAS Applications that do not have the resources to acquire large SOA implementations can be on-ramped onto the shared services, eliminating the need to develop their own Service Container box. Additionally, as more data becomes available, the more SIPs will become more creative on how they can create new forms of automation through the use of that information. This

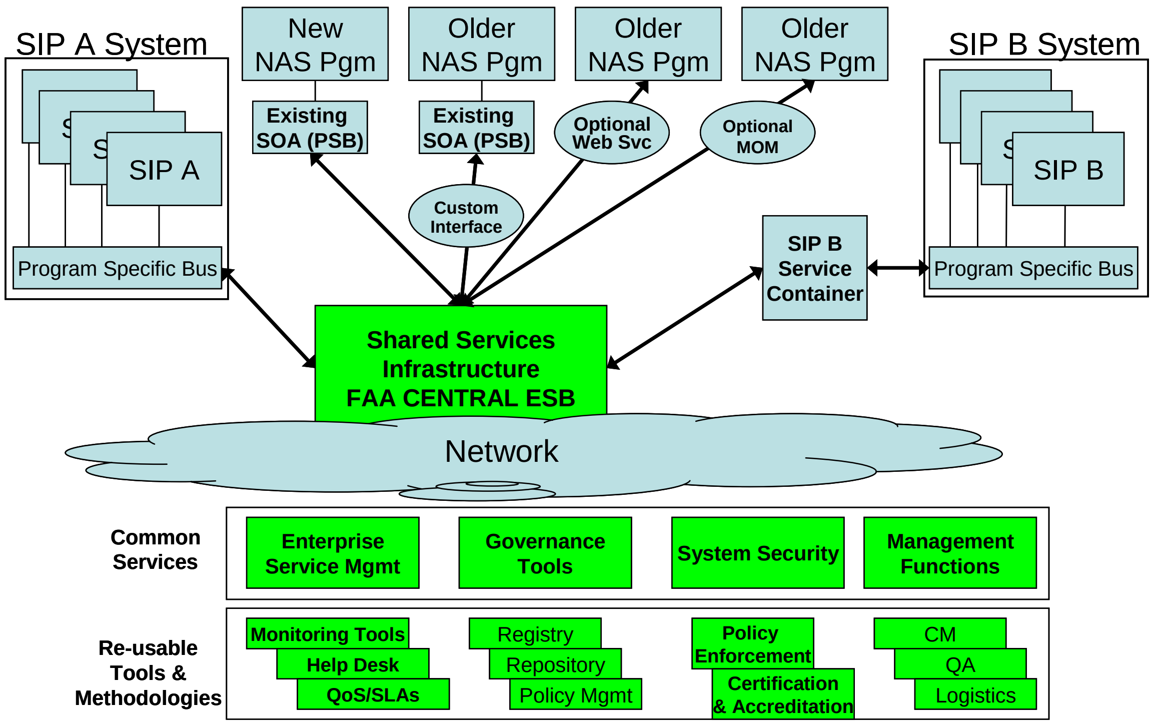


Figure 13: Central ESB, providing a NAS-wide message exchange

natural progression reinforces the earlier reference that the Central ESB will require considerable governance applied by the SWIM Program Office. New Applications targeting a SOA implementation utilizing a PSB, or older legacy applications with existing PSBs can choose from any of the multitude of standardized interfaces to integrate to the Central ESB. Legacy NAS Applications with their own PSBs, or existing systems can choose to integrate to the Central ESB via their own MOM, web services, or custom interface, giving them the flexibility to trade deployment method, cost and complexity that best fits their program needs.

For NAS application programs that seek to move forward ahead of SWIM Segment 2, there is real potential for building out their own SOA implementation, independent of SWIM. Without a strong governance concept and business policy definition/enforcement, the FAA faces the risk of disparate programs building out their own data exchange capabilities. Hence, although the technology enables greater collaboration between NAS Applications, separate interface management, messaging and service management infrastructure, creates another form of stove pipe; hence, SOA is introduced to the NAS as a new tool, but constrained by the same old business process, yielding a less efficient (optimized) IT and SWIM Infrastructure, and NAS.

The difference between the models presented here and the current SWIM Segment 1 Service Container model is in the deployment approach. The SWIM Segment 1 Service Container could potentially create new stovepipes if implemented by every new SWIM-enabled application being developed. The Central ESB illustrates how a local ESB's for larger SIPS and the consolidated ESB's for smaller SIPs offers the ability to evolve from point-to-point operations to a point-to-multi-point paradigm, where information and services are published once and subscribed by multiple users, optimizing re-use. The shared services strategy creates the foundation information exchange platform that leads to increasingly automated information products available for decision makers. The net result in the NAS environment, however, is improved decision-making, leading to improved NAS throughput, capacity, utilization and achieving the NextGen[6] objective of "reduced flight times".

2.3.7 Benefits of Shared Services

A service oriented interface provided by an automation system is defined in terms of a service provided (e.g. transaction request or subscription) using an open standard, without dependencies on vendor specific middleware or automation infrastructure, and supports multiple client components or systems without any software dependencies on the client system. These types of interfaces offer several advantages:

- Reduced development cost for interoperability - new client systems or components may be added without development impact to service based automation;
- Accelerated deployment of decision support tools - external automation tools supporting new operational procedures (e.g. CDA and RNP approaches) may be developed, tested and deployed independently from changes of the underlying automation systems;
- Information sharing for improved efficiency – distribution of surveillance and traffic to airlines and airport operators enables user decisions that improve capacity, efficiency and safety (user preferred flight profile, user specified prioritization of flights, user initiated en-route speed reductions for fuel savings).

A shared service is service implementation that may be reused between programs. Shared services are categorized as domain specific or enterprise services. Domain specific services are shared only within the domain/program that hosts them while enterprise services are shared among the enterprise, regardless of where they are hosted. Shared services are further characterized as business services and infrastructure services. Business services provide NAS domain specific capabilities, such as, generation of a weather product. Infrastructure services provide domain independent capabilities, such as, messaging, security and registries that enable information sharing among business services. Any of these shared services can be deployed using a geographically distributed or local deployment approach. Services that deployed in a geographically distributed fashion are subsequently referred to as distributed services in this paper. Services that are deployed in a geographically local fashion are subsequently referred to as local services in this paper.

Benefit of local services

- o Reduced latency for service invocation when deployed in geographically local fashion

- Increased availability of service
- Reduced development and maintenance cost for shared implementation
- Reduced operating cost for communications (reduced bandwidth)

Benefit of enterprise services

- Reduced O&M and travel cost for remote facility hardware
- Reduced license cost for remote facilities

Shared services provide the benefit of lower development cost through component reuse while also providing the flexibility of either local or enterprise deployment. For those implementing programs with specialized needs, the shared services approach allows for unique (and redundant) service implementation while maintaining interoperability between systems.

3 Segment 2 Structure and Needs

The following sections characterize the data types and services, drawing the distinction between business and infrastructure services, as well as the architectural impact of these services on the NAS.

3.1 Data Types and Services

As the NAS enterprise matures its understanding of SOA and web services, a migration of data and data types will naturally occur from a tightly coupled application/data structure to loosely coupled services and data. This section will explore some example data types and services that will likely be of interest for migration in the Segment 1 and Segment 2 timeframe. Many of these services will be implemented either as domain specific services (those that would reside in a Program Specific Enterprise Service Bus – PSB) or enterprise service (those provided by a Consolidated Enterprise Service Bus) where efficiencies are obvious or indicated. As the NAS Enterprise matures, many of the PSB-Based services will likely migrate to Consolidated Enterprise Services in later segments' timeframes.

3.1.1 Business Services

Business services provide NAS domain specific capabilities such as generation of a weather product or the exposure of flight data.

3.1.1.1 Flight Data

Some of the earliest services to be employed in the NAS (in ERAM Release 2) will be the exposure of Flight Data through the Flight Information Service. Initially this service will be a rudimentary capability that allows users to create, delete or update a Flight Object – the fundamental data type that represents the virtual existence of a flight in the NAS. Later ERAM releases will field a PSB based publish and subscribe mechanism that will make the Flight Object available to a wider array of users, thus increasing the situational awareness of the NAS.

3.1.1.2 Track Data

Aircraft position data as determined by automation systems will be made available on publish/subscribe basis to authorized users and will replace many of the point-to-point connections used today for this capability. Systems like the Traffic Flow Management Modernization (TFM-M) and the Traffic Situation Display (TSD) system that rely on aggregated Track data will benefit from this service. Point-outs and other track coordination tasks will be offered through this service.

Likewise, interagency coordination (between DHS, DoD, and FAA) will also be enhanced by services offered through the Track Data Service. Just as the FAA uses Intra- and Inter-facility point puts for coordination of traffic, an enhanced inter-agency capability will likely be made available that increases the accuracy and speed of coordination in a security event.

3.1.1.3 Traffic Flow Management Data

The Traffic Flow Management service, due to its NAS-wide nature, will be implemented as an Enterprise Service.

3.1.1.4 Sector Information

The PSB-based Sector Information service will provide visibility into a facility's sectorization at any given moment. Sectors are dynamically adjusted based on traffic density and occasionally equipment realities. The Sector Information service will expose the sector status and pertinent information describing the sector

3.1.1.5 Route Status Information

Arrival and Departures are often managed through pre-defined routes called STARS and SIDS (respectively). Whether or not these routes are applied to a flight plan is based upon an airport's runway configuration and which runway is in use. Adjacent and far facilities and many aircraft operators can benefit from knowing which arrival/departure routes are being used at a given time. This PSB-Based service will provide information on preferred route status use in a publish/subscribe mechanism.

3.1.1.6 Special Activity Airspace Information

Throughout the NAS are several types of Special Activity Airspace that must be managed and whose status must be made available widely and quickly in order to gain efficiencies in the system. Included are Restricted Areas, Military Operations Areas, Warning Areas, Alert Areas, Temporary Flight Restricted Areas, National Security Areas, and other such airspace. Most of these areas have published schedules that indicate when they are active as well as their three-dimensional boundaries. However, the controlling agencies of these chunks of airspace may not need them according to their published schedule, and returning the airspace back to NAS managers is a high priority.

The SAA service and data types made available through SWIM will aid airspace managers to quickly disseminate changes in the status of airspace that could then be used by non-participating aircraft and significantly shorten a planned route, saving fuel and time, while opening up additional airspace capacity. This will be introduced as PSB-based service and will likely migrate to a service residing on a Consolidate Enterprise Service Bus in the future.

3.1.1.7 General Information

The General Information Service provides a means to exchange general information/free text remarks with interfacing systems. The General Information Service is a peer-to-peer service exposed as a PSB-Based service. This service is envisioned as being a means to provide quick and direct communication between NAS entities that do not require the immediacy of voice communication and do not fall into the category of exchange of control data using the standard message interfaces between NAS systems.

3.1.1.8 Weather Information

Weather information made available through SWIM will likely be extensive and due to it's nature, may be implemented as both PSB_based and Consolidate Enterprise services. This type of data lends itself well to the publish subscribe mechanism, although data like Altimeter Settings, Runway Visual Range, and SIGMETS will likely have services associated with them that will allow create, update, and delete for authorized users.

Entire systems like the Integrated Terminal Weather System, Corridor Integrated Weather System, and Weather and Radar Processor will be able to publish large amounts of useful data – some system like WARP will also have separate client applications that will coordinate with back-end Application Server functions to further increase the data utility. NADIN services currently provided through the WMSCR system will likely be made available as Consolidated Enterprise Services.

3.1.1.9 Restriction Information

The Restriction Information Service provides a means to query or subscribe to the status, activation type and schedule for altitude and speed restrictions. ARTCC altitude and speed restrictions are based on Letter of Agreement (LOA) airspace constraints and Standard Operating Procedure (SOP) airspace constraints. These agreements are enacted to improve the efficiency of Air Traffic Operations and ease the workload of high density sectors. Knowing the specifics about a given restriction and making that data available to a wider set of authorized users will increase the utility of these long-standing agreements and improve the efficiency of the NAS. These PSB-based services will be available through publish/subscribe mechanisms.

3.1.1.10 Beacon Code

The Beacon Code Service provides a means to query or subscribe to beacon code reassignment and utilization information. Implemented as a PSB-Based service it will expose the Beacon Code information to allow air traffic managers to apportion and assign codes in an efficient and effective manner.

3.1.2 Infrastructure Services

Infrastructure Services are domain independent services potentially sharable by all domains.

3.1.2.1 Discovery

The Discovery Service provides design and run time capabilities for managing and accessing the service portfolio of the enterprise. A Registry provides the ability to make dynamic run-time connections to service endpoints enabling location transparent services. A Service Repository provides end-to-end lifecycle management of an enterprises service portfolio.

3.1.2.2 Service Level Security

The Service Level Security service provides message and service level access protection. Identity management provides the ability to assign identities and credentials to users and identities to SWIM equipment in support of access control. Access management determines, based on the client's credentials and the current enforcement policies, whether access will be granted to a service or message.

3.1.2.3 Service Management

The Service Management Service monitors and manages SOA service access points and delivery points. It monitors and enforces SLAs associated with NAS business transactions and monitors and manages middleware components as well as the servers that host the middleware components. It also provides and manages transaction audit trails.
Enterprise management is responsible for monitoring the health and performance, controlling the configuration and managing faults at all levels of the enterprise, from network elements up to business processes

3.1.2.4 Enterprise Service Bus Service

The Enterprise Service Bus (ESB) Service provides a common backbone upon which global information is transacted and managed. This ESB provides a unified, approach to enforcing service levels, and a common means of applying service-level security and service management policies within a service infrastructure. It also provides extensive mediation services for protocol and message payload transformations enabling interoperability of technology disparate systems.

3.2 Architectural Impacts NAS & Non-NAS Systems (JPDO View)

3.2.1 <u>Locations and Capabilities/Needs</u>

If the SWIM program is deployed correctly, it is unlikely that the typical Air Traffic Controller or Supervisor will see any change at all. That is a significant and far-reaching observation. If the user will not directly notice a change in the operations of the NAS, why is SWIM being introduced?

The answer lies in how SWIM will allow the NAS automation systems to continue to perform as they do now with extended reach of data.

This section will briefly discuss the implications of making existing data available to authorized, non-traditional users and systems and how the entire NAS will benefit from advancements in a few facilities.

3.2.1.1 En Route NAS

The Air Route Traffic Control Centers (ARTCC) of the NAS (En Route Centers) are, hierarchically speaking, the center of data activity for a given flight. As a flight progresses through time and space and is passed from center to center, the ownership of the flight object – the abstract representation of the flight in the automation system – also passes. The flight object holds all knowledge of the history and future of a flight including its present state. It is easy to see why access to this object is one of the first SWIM services to be deployed.

By making available the flight object and its contents to the greater NAS community (to include the airlines, DoD, DHS, and international Air Navigation Service Providers) SWIM will throw open the shutters to the state and intention of a given flight; thereby, allowing appropriate systems and planners access to the very information that is crucial to optimizing and securing the airspace.

Because of the central role that the ARTCC and its data play in the NAS, it is likely that from an architectural viewpoint, ARTCC facilities will deploy and operate a Program Specific Enterprise Service Bus (PSB) suite of hardware and software that will ensure the efficiency and security of the data. The transition from present state (limited SWIM) to a final future state (full SWIM) will be determined largely by the needs of the NAS and the ability for the en route automation systems to make use of SWIM and its capabilities. Whether this involves a suite of hardware and software per facility, a single suite of hardware and software for all facilities, or some combination of these paradigms will depend on the outcome of performance testing, modeling and simulation, and other empirical methods of data gathering that will determine how best to deploy.

3.2.1.2 Terminal Facilities

Terminal facilities provide a pathway for the transition between en route operations and the airport environment (landings and takeoffs), and serve a vital role in separation of traffic in highly dynamic airspace. The flexibility to maneuver around weather is more restricted in the terminal airspace than in en route, simply because of the lower altitudes and the fixed location of airports. Lower altitudes also brings together a mix of aircraft and pilot capabilities that makes terminal a different challenge than en route.

Terminal facilities have been evolving towards combined metroplex architectures. That is, the bringing together of multiple facilities into a single building and leveraging common equipment and infrastructure between them. Sothern California, Potomac, and New York Terminal Radar Approach Controls (TRACONS) are examples of this trend.

The sophistication of these facilities and the equipment required likely means that they also will require a local suite of hardware and software to serve as their ESB. As was noted above, the ARTCC and its automation system will be the owner of the flight data object that represents a flight in the NAS. All other NAS facilities that require access to the Flight Object will use the Web Services interfaces defined in the Flight Information Service detailed in section 3.1.1.1 above.

There are also, however, several locations throughout the country where metroplex facilities are not needed or planned. In some of these small facilities a connection to another facility’s PSB or perhaps to the ESB hosted on the Wide Area Network (WAN) through FTI will provide the necessary connectivity and services. Again, testing of performance and simulation and modeling will be required to determine the optimum architecture to support this need.

3.2.1.3 Airports

The airport facility and associated Air Traffic Control Tower (ATCT) are some of the most numerous facilities in the NAS. Large airports that service part 121 operations number near 500, and smaller airports serving general aviation and specialty operations raise that number above 10,000. It is highly probable that some of these airports (notably the largest in the country) will make use of large suites of equipment to meet their needs and will have the technical staff and power/cooling capabilities to house their own PSB. This would be a rare occurrence, though.

Most likely, airports and smaller facilities would have their SWIM needs met by utilizing a WAN-based ESB or possibly by leveraging a PSB from their controlling TRACON or ARTCC.

The data supplied by airports would be relevant to airport operations: RVR, winds, local altimeter setting, and other meteorological data are prime examples of data that the airport would likely publish.

Surface surveillance data is also likely to be made available through SWIM publish/subscribe mechanisms, providing a more accurate position of aircraft between the gate and departing or arriving runway. The airlines would likely have a great interest in this data and would be the main subscribers.

3.2.2 Enterprise Services (Global)

Enterprise Services provide capabilities that can be leveraged across the enterprise by multiple service domains avoiding duplication of services and the associated increased development and operational support costs. This section describes candidate Enterprise Services that are required NAS wide, methods of making those services available across disparate NAS users.   Other considerations for monitoring services, establishing service level agreements and ensuring an appropriate security level is established, are included in this section.

3.2.2.1 Discovery (Interface Management)

Discovery services cover a broad set of functionality from helping make dynamic run-time connections to service endpoints to managed end-to-end lifecycle of a service.

3.2.2.1.1 Use of repository for lifecycle management of services

An enterprise implementing SOA must manage a portfolio of services. This includes planned services, those in development, deployed services and services being retired or transitioned. The repository provides different views of the service portfolio to users with different roles. Business analysts need to know what business processes the service supports, who is responsible for development and support of the service, how the development is funded, etc. The software architect need to know what standards and technologies the services require, which architectural layer they belong to, their dependencies, and how they conform to the enterprise information model. Developers, operations and systems engineering represent additional roles.

The repository also manages how this information is changed through workflows that give key roles authority to review and approve changes. An example of this class of tool would be the Oracle Enterprise Repository (formerly AquaLogic Enterprise Repository) which provides highly extensible assets and lifecycle workflows. Other tools include WSRR from IBM, Systinet Repository (not UDDI) from HP, Repository Manager from SOA Software, and Galaxy from Mule.

Repositories are enterprise level tools for planning and development and are not meant to support critical run-time access. The repository should be able to utilize the stored metadata and support some aspects of dynamic provisioning of the services for example, pushing defined security policies to security management products that enforce policies at run-time. Since these tools support multi-domain architectures, they are able to segregate SIP specific service information from NAS wide service information. There is no compelling technical reason to have more than one repository. However there may be business reasons, such as organizational boundaries, for an enterprise to have multiple repositories.

3.2.2.1.2 Use of run-time registry for service location transparency

A registry (UDDI or other) provides a run-time source for information needed to connect with and use a service. UDDI is a popular standard which supports services described using WSDL and XSD. Though WSDL can be used to describe other kinds of services, primarily SOAP/HTTP and SOAP/JMS web services are widely used. Other registry technologies exist, but many other SOA services and tools depend on integration through a UDDI interface. Popular products are Systinet Registry from HP, and the open-source JUDDI (UDDI v2 only). Most vendors resell or integrate with the Systinet registry. Registries do not provide a true service lifecycle model, rich metadata about services or a strong concept of roles and views of a service.

3.2.2.1.3 Federation of repository and registry

While a repository is intended to span organizations and service lifecycle stages, the registry is intended to hold actual endpoints of operational services. Enterprises that need to expose different services to different groups will needs multiple registries. The promotion and exposure of services from development and test to operational registries and to global registries should be controlled by the repository via workflows. The Oracle Enterprise Repository provides a tool to promote services to the UDDI.

Some registries have a federation feature that enables multiple distributed registries to be managed as one logical registry. This feature can be leveraged to manage an enterprise wide service registry and the local domain registries in a more consolidated fashion reducing operational support and providing an automated approach to the synchronization of service registries.

3.2.2.2 Service Level Security

Authentication is the verification that a user, organization, system, document or other object is who, or what it claims to be. For users this is typically consists of a password challenge. The goal is to issue one challenge per session to authenticate the user system wide. This is commonly known as SSO (single sign on). Since only the user interface layer can prompt for a password, web services and others must rely on a token or certificate to provide the user's authentication. WS-Security provides a standard for calling a web service with a SAML token. Users and their passwords are maintained and checked in an LDAP directory service such as MS Active Directory or OpenLDAP Tokens are issues by an STS (token service) such as IBM TFIM. WS-Security and security propagation via SAML can be achieved by the SIPs via Artix Security which is being offered by the FAA SWIM.

Authorization is the process of granting or denying access to a resource based on the identity or role of the requestor and security access policies. These policies may be defined using the XACML standard, but often use proprietary formats. Because policies should be defined and maintained globally, but enforced closer to the resource, the PEP (policy enforcement point) is often separated from the PDP (policy decision point). Examples of PEPs are IBM WebSeal for protecting web applications (UI), and the IBM DataPower appliance for protecting web services. IBM Tivoli Access Manager (TAM) is an example of a PDP.

Privacy is provided through encryption. WS-Security provides for part of a message to be encrypted, allowing middleware to act on the other parts.

Non-repudiation means the ability attribute a message to its originator, and ensure that the content hasn't been changed. This is achieved through digital signing. Signing and encryption can be expensive, and are often supported by accelerators, such as, IBM DataPower.

Auditing requires that transactions are logged, and the logs identify the user that initiated the action. This requires the propagation of identify information through the entire processing chain, even across domains.

3.2.2.3 Service Management (Enterprise Management)

Service management is a subset of enterprise management. Enterprise management is responsible for monitoring the health and performance, controlling the configuration and managing faults at all levels of the enterprise, from network elements up to business processes. Service management focuses on monitoring of middle tier services, such as, web services, or JMS topics and queues.

In the past, these have not been part of the standard enterprise management world, but performance and faults of the supporting infrastructure impact services, and service performance and faults impact enterprise level functions. A key element to the successful monitoring of SOA services is the integration of the applications level monitoring with the network monitoring, enabling the correlation of network service outages with SOA service outages or degradations and vice versa. This expedites root cause analysis and determination leading to quicker problem resolution at lower cost.

Services must be explicitly monitored, because it is possible for the IT infrastructure to all be functioning nominally, but have users that can't get the data they need. Tools, such as, Actional, AmberPoint, Rio or IBM Tivoli Composite Application Manager (ITCAM), monitor web service performance against SLAs (service level agreements) and generate alerts when the SLAs are in violation. AmberPoint can also monitor non-web services such as JMS endpoints, and JDBC connections. This is analogous to the way FTI currently manages SLAs for network communication services; however, FTI's SLAs represent legally binding contractual constructs, and have associated financial penalties for SLA violations.

3.2.2.4 Enterprise Service Bus (Messaging)

While often discussed together, messaging (MOM) and ESBs are sometimes provided as separate functions. Some vendors package the MOM and ESB as one integrated product while others package them separately. ESBs allow service providers and consumers to connect with the ESB and, utilizing built-in mediation services, compensates for differences in message format and protocol, as well as providing routing, and often security and performance monitoring components. Many organizations find that they need a federated ESB pattern. This pattern typically includes one ESB at the enterprise level, and one ESB for each major domain area within the enterprise. Exchange of information within a domain is managed within the domain ESB while exchange of information external to the domain is managed by the enterprise ESB. This approach enables easier management of messaging within the enterprise, allows local domains to specialize the ESB to their unique requirements, and allows each domain to effectively establish domain specific governance. ESBs typically exchange information with each other using Web Services (HTTP/SOAP) and JMS. Less common exchange methods are access to JDBC data sources, files and legacy services such as CORBA.

While there is no special standard for inter-ESB communication, the JBI standard does provide for portable ESB connectors. A JBI compliant CORBA adaptor from one ESB product, can be added to a different ESB product to provide that functionality. While most ESBs come with a JMS, they can connect to any JMS by using the vendor supplied JMS provider (connection factory). Popular ESB products include AquaLogic Service Bus (now named Oracle Service Bus), IBM WebSphere Message Broker, Mule, Iona FUSE (Apache ServiceMix), and OpenESB (Sun/Glassfish).

3.2.3 <u>Communications</u>

This section provides the communication mechanisms for transporting services across the various NAS user communities. This section identifies and describes the application level communications mechanisms recommended for information exchange in SWIM Segment 2. Messaging APIs, transport mechanisms and standards for message transfers are described. Recommended message structure standards and applicable usages are discussed.

3.2.3.1 JMS

The Java Message Service (JMS) is Java Message Oriented Middleware (MOM) for reliable Enterprise messaging between two or more clients. JMS is a part of the Java Platform, Enterprise Edition, and is defined by a specification developed under the Java Community Process as JSR 914, and as such is a de facto industry standard. JMS is an Application

Programming Interface (API) specification (JSR914) that states how a client application is to interact with the underlying messaging system. There are no standards for over-the-wire (OTW) protocol for messaging systems; therefore, it is probable that each JMS-compliant messaging supplier is using completely different OTW protocols.

Enterprise messaging is very important since it provides a service for the exchange of critical business data and events throughout an enterprise like the FAA's envisioned NextGen enterprise. Messaging, like JMS, is a form of loosely coupled distributed communication, where loose coupling describes a design principle where integration interfaces are developed with few, if any, assumptions between the provider and consumer. This reduces the risk that a change in one endpoint will affect any other endpoint. This loose coupling of interfaces can be additionally and dramatically improved when providers message data using an adaptable format such as eXtensible Markup Language (XML).

JMS supports two basic types of communication models, point-to-point messaging and publish-subscribe messaging. In the point-to-point (queue) model, a provider posts messages to a consumer's queue and the consumer gets messages from the same queue. In point-to-point messaging, the provider knows the destination of the message and posts the message directly to the consumer's queue; thus, the provider knows about the consumer's queue interface, but nothing about the consumer. The publish/subscribe model supports publishing messages to a particular message topic. With publish/subscribe, multiple consumers can register interest in receiving messages on one or more particular message topics. In this model, neither the provider nor the consumer knows about each other. Both point-to-point and publish-subscribe have their place within an enterprise messaging system.

JMS, which is an enterprise message-oriented technology, greatly reduces tightly coupled communication, e.g. point-to-point, through the use of an intermediary component known as a queue. A queue allows providers and consumers to decouple themselves from each other. Since the providers and consumers communicate with the well-defined queue, providers and consumers need not know about each other and only have to maintain their interface to the queue. Typically, data communicated using a queue is put into a canonical, or standard, form. In that way, providers and consumers may change their internal communications as long as they remain compliant with the canonical form for enterprise data messaging.

As with most software components, JMS MOM products can be obtained through open-source and proprietary (private or closed-source) providers. Depending on the particular product offering, the MOM may be bundled with other SOA components or may be provided independent of other SOA components. Software architects and developers can choose from a variety of open-source and closed-source suppliers of a MOM.

There are several major open-source MOM products available today; and, there are additional low-market-share products available, too. Following is a non-exhaustive, alphabetical list of open-source JMS MOM products[7].

Opensource JMS MOM[8]	Supplier
▪ ActiveMQ	**Apache (Iona)**
• Java Open Reliable Asynchronous Messaging	**ObjectWeb**
• JBoss Messaging[9]	**Red Hat**
• MantaRay	**Coridan Technologies**
• Open Message Queue[10]	**Sun Microsystems**
• OpenJMS	**OpenJMS Group**
• **Rabbit MQ**	**RabbitMQ**

Table 1: Open-source JMS MOM products

Some open-source MOM products are supported entirely by voluntary communities, e.g. ActiveMQ, while other open-source MOMs are supported by vendors offering maintenance contracts. Typically, with the vendor-backed open-source MOMs, the product user is indemnified for any Unix/Linux copyright infringement and has a product with a known pedigree.

There are several major closed-source MOM products available today. Following is a non-exhaustive, alphabetical list of closed-source JMS MOM products[11].

Closedsource JMS MOM[12]	**Supplier**
• Advanced Queuing (AQ)[13]	**Oracle**
• EMS	**Tibco**
• Fiorano MQ	**Fiorano**
• iBus	**Softwired**
• JMS	**Sun Microsystems**
• JMS Grid:	**Sun Microsystems**
• Microsoft Message Queuing (MSMQ)[14]	**Microsoft**
• Oracle Application Server JMS	**Oracle**
• SonicMQ	**Progress**
• SwiftMQ	**IIT**
• WebLogic Server JMS	**BEA Systems**
• WebSphere MQ[15]	**IBM**
• **Windows Communication Foundation (WCF)[16]**	**Microsoft**

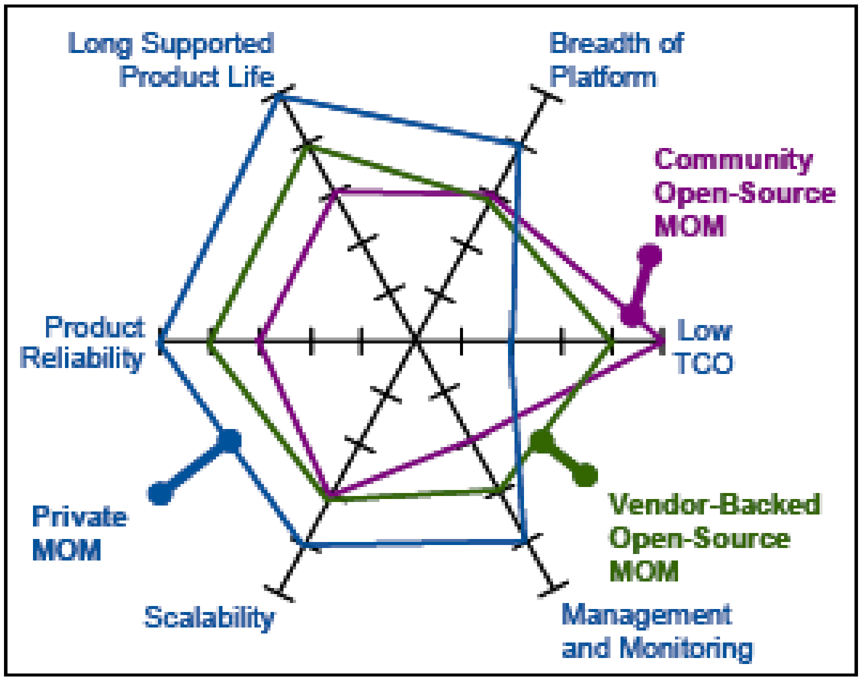
Table 2: Closed-source JMS MOM products

There are clearly advantages and disadvantages of choosing open-source or closed-source JMS MOM products. Figure 9 represents an analysis performed by Gartner[17]. Some salient facts as a result of analysis of the figure, comparing closed-source and vendor-backed open-source, are summarized as follows.

1. Open-source has better “TCO” than does closed-source, typically since the closed-source includes license, support and technical support staffing costs and open-source is based on community open-source with none of these services available.
2. Closed-source has better “breadth of platform” than open-source; typically due to open-source concentrating on one or only a few operating systems and languages.
3. Closed-source has better “long supported product life” than open-source; typically due to the revenue gained by the vendor for supporting older product versions.
4. Closed-source has better “product reliability” than does open-source; typically due to the rigorous and exhaustive verification (testing) performed by the vendor and the low number of defects.

5. Closed-source has a better "scalability" than does open-source; typically due to the vendor's support of a large number of messages per day and a desire to support a wide range of deployments.
6. Closed-source has better "”management and monitoring" than does open-source; typically due to the closed-source integrated tools to complement the MOM .

In summary, open source drives TCO, while all of the technical factors favor open source (according to Gartner). However utilizing the recommended shared services model, where common services are shared amongst all participating programs, will drive down the overall TCO for the FAA; hence, they realize all of the value-benefits derived from the technical parameters of this trade matrix, while spreading the cost to achieve an improved TCO. In summary, as a result of the referenced report, the only potential advantage of open-source over closed-source is TCO.



Source: Gartner (March 2008)

<u>**LEGEND**</u>
Closedsource (Private) MOM
Community Opensource MOM
Vendor-Backed Opensource MOM

Figure 14 Comparing the Best Open-source and Closed-source JMS MOM Products

With the large collection of messaging products available, it is inevitable that some of these products have already found their way into NAS systems. Some of the messaging tools will be JMS compliant (JSR914) which will significantly reduce the risk of interoperability between heterogeneous messaging systems. Other messaging tools may be non-compliant to JSR914, using proprietary implementations of a messaging service, and not offering a JMS interface. It is the proprietary message implementations which should cause pause.

It is worthy to note that some of the more mature messaging infrastructures have provided non-JMS API's for a long time and legacy implemementations based on these non-JMS API's would need to be integrated with with JMS based solutions. At the same time, most of these mature messaging infrastructures now support the JMS standards and can be used to provide new solutions that are both standards based and provide that advantages of maturity.

Many legacy applications were not built with the message model in mind; therefore, it may be the responsibility of the ESB to transform the messages into a legacy format that is understandable by another application. The ESB may use its own messaging infrastructure or it may provide adapters (e.g. JMS, WS-Rel*) to interface to any of the popular messaging infrastructures (e.g. WSMQ, MSMQ, Tibco Rendezvous). As with any integration solution, the collection of adapters supplied by an ESB supplier varies; however, most of the major messaging infrastructures and applications are addressed.

ESBs can be either single protocol or multi-protocol, referring to how to gain access to the ESB. Most single-protocol ESBs require a WS-SOAP interface, but to send an XML message using JMS, an adapter translates the incoming message to SOAP to participate on the bus. In a multi-protocol ESB, varieties of protocols (e.g. WS-SOAP, JMS, JCA) natively interact with the bus, without employing an adapter.

Through the use of JMS, it is possible to transmit data formats of many different types, including binary, without the need for convoluted transformations. JMS is used for publish-subscribe, which will be the prevalent communication model used within NextGen (rather than request/response). JMS can also support many different message exchange patterns (MEP) to provide the developer with a rich suite of messaging capabilities, including variations on asynchronous and synchronous messaging.

3.2.3.2 SOAP

SOAP is an XML-based protocol, standardized by W3C, to provide applications with the capability to exchange information, normally using http, but supported by other transport protocols as well. SOAP standardizes the practice of using XML and http to invoke methods across the internet. SOAP is a mechanism for an application running on one platform type, such as Sun Solaris, to communicate with an application using the same or different platform type, such as Windows, by using http and XML as the mechanisms for information exchange. SOAP is independent of the host platform, supporting languages such as Java, C#, and Perl; and, supporting operating systems such as Unix, Linux, and Windows. SOAP brings other advantages as well: using SOAP over http eases communication through firewalls; and, SOAP can use transport protocols other than http, such as JMS, Tuxedo, MQSeries, Tibco, etc.

Web protocols are available for all major operating systems; therefore, http and XML provide a solution for applications hosted on the same or different platforms to communicate. SOAP specifies exactly how to encode an http header and an XML file so that an application on one platform can invoke an application on another platform and transfer data. Additionally, SOAP specifies how the server application can return a response. SOAP is now commonly used as a protocol to invoke a web service.

A SOAP message may need to be transmitted together with attachments of various sorts which may be textual and/or binary. A SOAP message must conform to XML rules for allowed characters and character sequences; therefore, binary data can not be included in the XML message. Additionally, SOAP implementations typically parse the entire SOAP message before deciding what to do with the contents, so large data sets could overwhelm the platform’s resources. SOAP provides a mechanism for transporting large payloads and binary data as an attachment, without violating XML rules. The solution to this problem is to create an opaque payload using SOAP with Attachments API for Java (SAAJ).

SOAP with Attachments extends a SOAP message to include, in addition to the regular SOAP part, zero or more attachments. Each attachment is defined by a MIME type and can contain any content represented as a byte stream. MIME, or Multipurpose Internet Mail Extensions, first found its use in email systems where it was necessary to support 8-bit ASCII and non-text characters. MIME uses base64 encoding to encode arbitrary octet sequences into a form that satisfies the rules of SOAP.

3.2.3.3 MTOM

SOAP Message Transmission Optimization Mechanism (MTOM) is used to optimize the transmission of XML Information Sets (XML Infosets[18]) containing non-XML content, i.e. binary data. Software designers often want to use the structured, extensible markup conventions of XML without discarding existing data formats (legacy) which are not compatible with XML syntax. Unfortunately, XML is not the ideal carrier for binary data. XML uses a text format, and as such doesn't deal well with binary data. This means leaving the existing non-XML formats as is, to be treated as opaque sequences of octets by XML tools and infrastructure. Binary data may be properly encoded, using base64Binary, with the result that XML can contain binary data. Such an approach allows legacy NAS data formats to be encoded for use with XML.

Unfortunately, base64 encoded data expands by a factor of 1.33x its original size[19]. There is also the overhead and latency in resource utilization for this format, especially when decoding back into raw binary. These performance concerns have over-shadowed using embedded data in XML.

With SOAP, XML schemas can be used to specify new data types and those new types can be easily represented in XML as part of a SOAP payload. It is because of this integration with XML Schema that data types can be encoded in a SOAP message.

In an XML message, it is required to convert binary information using base64 encoding. MTOM combines the composability of base64 encoding with the transport efficiency of SOAP with Attachments[20] (SWA). MTOM provides more efficient encoding of binary data in a SOAP request or response. It uses XOP[21] (XML-binary Optimized Packaging), which specifies the method for serializing XML Infosets with non-XML content into MIME packages for the transmission of binary data. MTOM enables SOAP to optimize the transport of a SOAP message by selectively encoding portions of the message.

To provide the ability for a NAS system to transmit binary data efficiently between web services, it will be ideal to use SOAP MTOM. This will provide the highest current efficiencies for transferring binary data using the SOAP protocol.

3.2.3.4 Legacy

There will be many legacy NAS systems which can be modified or adapted to interoperate with the NAS NextGen SOA. Although many of their current interfaces will not be modified or redirected in any way, it will be possible to modify some aspects of their current operation or provide mediation that will permit legacy applications to take full advantage of the capabilities provided by SWIM. Through use of the ESB, adapters, and format mediation, it will be possible to convert the legacy data into any format desired for the SWIM participants. Although mediation can occur anywhere, it is recommended that many adaptations that do not require considerable domain specific business logic be performed external to the legacy system. This approach minimizes changes and associated impacts to legacy systems.

With the FAA working to define and adopt XML for NextGen message formats, many of the legacy systems can have their data mediated into a canonical XML form, using such methods as Extensible Stylesheet Language Transformations (XSLT) or XQuery, both of which use XPath as a sub-language. XQuery's capabilities overlap with XSLT's; with both providing the capability to directly manipulate XML data. XSLT is an XML-based language used to convert data between different XML schemas. With XSLT, the original XML document is not changed; a new XML document is created based on the content of the existing document. XQuery provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML. It is semantically similar to Structured Query Language (SQL), and provides a syntax which provides for the creation of new XML documents. XQuery is a simpler language than XSLT to learn and use, but it sacrifices usability for capability, lacking such mechanisms as polymorphism. Each serves their purpose and can be effective message transformation mechanisms for SWIM.

Additionally, there will be legacy systems for which there will be a limited capability to expose their information to the NextGen community. Their platform resources or software development environment may not support modification of their legacy software or the addition of adapters. In this case, an external platform may be employed to capture the legacy system's communications and perform mediation in order for the legacy system to remain unaffected. The following legacy systems are natively supported by the FAA's SWIM service container:

- C/C++
- Tuxedo
- TIBCO
- Java
- RPC
- .NET
- WebSphere MQ the native interface (MQ also supports JMS)
- CORBA

This whitepaper has discussed some of the options available for mediation which will greatly assist in SWIM-enabling legacy systems. Based on the wide variety of systems and applications within the NAS, each will require examination to determine the most suitable method for connecting them to SWIM for interoperability throughout the NAS.

3.2.3.5 Non-FAA Communications

Air Traffic Services are governed by the International Civil Aviation Organization (ICAO) Convention under which states commit to providing ATS services in their airspace over their territory and the surrounding oceans.

On a day to day basis civil aviation authorities communicate with each other and with airlines and airports for the exchange of operational messages such as flight plans, weather messages etc.

The current legacy messages are exchanged through AFTN (Aeronautical Fixed Telecommunication Network). AFTN is created by the Air Traffic services providers, following ICAO standards, and has a messaging address structure based on ICAO codes. It now comprises some 150 national networks with connections to their national terminals and those in adjacent countries. AFTN traffic consists of NOTAMS, Flight Plans and Slot Allocations, and operational meteorological data. It circulates on the internal AFTN network, between different ATS providers, and between providers and airlines.

A new generation of ICAO messaging referred to as the Aeronautical Message Handling System (AMHS) was finalized in 2002 as part of the ICAO standard for the Aeronautical Telecommunications Network (ATN). The AMHS uses a set of protocols derived from the ITU X.400 standard. AMHS systems can interconnect using the X.25 protocol, or the planned ATN networks that will use OSI protocols or to use AMHS over TCP/IP using RFC1006. The deployment of AMHS is slow due to its questionable aging technology.

While AFTN messaging has served the ICAO community well, it is not extensible to meet the needs of more demanding newer applications with much more complex data. Additionally it remains an aging technology that requires serious replacement. AMHS which is based on X.400 has been recommended to replace AFTN. However it remains yet another outdating technology with high cost of deployment, scarce expertise and products. And while there are AFTN and AMHS approaches that leverage the benefits of IP networks, current implementations in the industry are still constrained to using costly network technologies such as X.25 or CLNP routers.

Airline exchanges are governed by IATA/ATA standards. The current industry reliable messaging for business critical exchanges use an IATA standard referred to as TypeB. Gateways are also developed to bridge AFTN to TypeB and consequently enable seamless message communications between ATSOs and airlines.

Leveraging open standards such as XML based messaging and using the Web Services communication framework has the potential to transform how business class messaging is accomplished in the industry through new technology use. For air transport operational messaging a new standard called TypeX is under finalization with the related IATA Work Group that specifies the use of industry standard addresses and compliant with IATA and ICAO codes is emerging to support XML messaging and industry business practices. TypeX has been implemented by SITA and ARINC and some users. TypeX is SOAP based and can be used in a SOA environment and plugged into ESB. TypeX makes use WS-* security specifications for security functions

The use of TypeX by FAA for communication with the other civil aviation authorities and airlines meets the directions defined within the scope of SWIM. It facilitates communications by the use of a single protocol that meets ICAO and IATA messaging and addressing requirements while making use of a widely supported open technology. Communications with AFTN and airlines TypeB is facilitated by gateways that are simple to build due to similarity in addressing and some other industry related technical principles.

3.2.4 Leveraging the SOA Best Practices and Governance White papers

REDACTED:

The team had originally envisioned establishing a section that captured the elements from the previous GEIA bodies of work from the SOA Best Practices & Governance White Papers. Instead of having a consolidated section of references to these works, these references are identified throughout, in text. For example, Section 2.2 presents a SOA Reference Model originally presented in the SOA Best Practices White Paper, as a frame of reference. These references demonstrate how these previous works lend themselves for use within the context of transitioning from Segment 1 and Segment 2.

3.2.5 Industry Best Practices

There is an extensive body of knowledge around this mature technology. The following section provides Commercial SOA Standards & Best Practices, and references the Practical Guide to Federal SOA.

3.2.5.1 Commercial Standards & Best Practices

The SOA paradigm is over a decade old and there is a large, public body of knowledge available on the internet. There are several, excellent, websites dedicated to SOA that are focused on interoperability standards, security and the definition of next-generation SOA concepts (Service Component Architecture and Service Data Objects). Finally, a hallmark of the maturity of a discipline is the existence of a pattern library. A community effort started several years ago to create just such a library of design patterns, equal in stature in the software engineering community to both the well-known Gang-of-Four book and Gregor Hohpe's book of EAI patterns. The following four resources were contributed to by dozens of SOA practitioners over the past decade and should be referred to for their wealth of knowledge and command of the subject:

Organization for the Advancement of Structured Information Standards (OASIS)
http://www.oasis-open.org/home/index.php

Open Web Application Security Project (OWASP) http://www.owasp.org/index.php/Main_Page

Open Service Oriented Architecture (OSOA)
http://www.osoa.org/display/Main/Home

SOA Patterns Community
http://www.soapatterns.org/

For insight into how standards are evolving and current practices that will eventually be incorporated into the above bodies of knowledge, a handful of publications and industry thought leaders' blogs should periodically be referred to. Linthicum, Biske and Governor are widely recognized to be the most reliable "information radiators" regarding SOA. Additionally, both SOA Magazine and the Architecture Journal provide excellent snapshots of the state of the art.

We would like to emphasize that the art and science of SOA is continually evolving and growing. The collection of resources we have provided substantially supplement what is freely available from the collection of vendors selling product into the SOA marketplace. We therefore encourage readers to refer to these resources, frequently.

3.2.5.2 Federal SOA Practical Guide

The introduction of Service Oriented Architecture requires changes that are not all technical in nature. When such a fundamental transformation to the tenants of engineering occurs a management champion at the highest levels must be recruited to carry the message of change to the entire organization. And the message of change itself should contain an imperative that makes the need for the change obvious. Through this high-level sponsorship and message, each organization in the enterprise will see how its function and output can best address or be addressed through the change. For the FAA, this is no different.

As is well known, the NAS is a collection of individually developed systems that have their roots in unique circumstances: each one developed according to a specific need and with discrete interfaces in mind. The recognition that this tightly coupled grouping of systems has more to offer the enterprise is the key message of SWIM. The demonstrated vision of NextGen – with SWIM as an enabler – shows the leadership of the FAA is embracing the change necessary for success.

Through the governance of the SWIM system the FAA will have a unique opportunity to leverage the technical changes driven by the introduction of SOA into the NAS organizational and decision making structure. According to the *Practical Guide to Federal Service Oriented Architecture Version 1.1* Federal IT governance is defined as

> …coordinated decision making involving both the center [central authority] and the business units," suggesting a two tiered vertical model that shares power in some manner[22]

Through a mechanism like this decisions can be made that benefit the operational entities of the NAS while maintaining the integrity of the SWIM model.

The success of SWIM will be directly related to how well the existing organizational and decision making structures evolve to a cross-program mindset that develops and introduces services into the enterprise, requiring a blurring of the traditional lines of program management structure. As was posited in the *GEIA White Paper on SOA Governance Best Practices*[23]

> Service orientation demands even higher stakeholder involvement and coordination as the organization tries to create business services, used throughout the enterprise with services, to collaborate and participate in multiple interdependent processes[24]

Typically, the nature of service development in a SOA construct leads to data aggregation across systems in ways that were not foreseen when the systems producing the data were procured. Early in the SWIM program's development, a governance construct that guides and aids the legacy program offices in crossing boundaries and breaking down barriers to innovation must be in place. Without this guidance, program offices throughout the FAA will have little incentive and no precedent for evolving the NAS into the next generation of Air Traffic Control.

3.3 Potential Difficulties Implementing SOA in the NAS

Describe possible difficulties that FAA could encounter which could be examined through trade studies, white papers and recommendations based on industry best practices.

3.3.1 Performance of Messaging Core Services over WAN

There are two key issues that have the potential to impact the performance of messaging core services over a WAN. First, the decisions regarding the representation of data, in particular utilizing XML can result in increased message sizes. Second, the routing strategies employed to effectively deliver data can minimize WAN bandwidth utilization.

3.3.1.1 Data Representation

A common concern about implementing SOA architectures is the standardization on XML representation of data and the resulting increased message sizes leading to increased network bandwidth consumption and decreased messaging throughput performance. However, this may not actually be the end result for the NAS for several reasons.

SOA implementations often use SOAP/HTTP with XLM representation. This often creates the perception that SOA architectures require XML representations but this is not the case. The SOA methodology does not specify or require a specific representation.

Not all shared information is a candidate for XML representation. For example, many weather products are currently represented in binary array formats and there are better self describing standards such as HDF5 that can be utilized for this type of data that do not lead to significant expansion of the data. This type of data would only need to be wrapped with an XML header for routing purposes which would result in only a small percentage increase in overall message size.

For transmitting binary data via SOAP, the MTOM standard can be utilized to efficiently transmit the binary data without having to convert to XML. Existing textual data would be a much more likely candidate for XML representation. All planned information exchanges need to be analyzed on a case by case basis to determine whether XML representation of the information is advantageous to anticipated consumers as weighed against potential network bandwidth utilization increases. Consumer usage patterns need to be evaluated in making this determination.

3.3.1.2 Routing

With the introduction of ESB enabled content based routing, messages can be intelligently routed at a finer granularity to consumers resulting in decreased overall network bandwidth utilization which could potentially more than compensate for increased message sizes due to XML expansion. For example, several NAS systems currently receive streamed weather data that includes all weather products when in fact only select weather products are actually needed. With content based routing it becomes possible to route only the desired products to these systems.

In order to accurately estimate the impact to messaging throughput and network bandwidth utilization, a study should be performed that identifies and characterizes the information to be exchanged among domains, determines the anticipated providers and consumers, and defines the optimal message representation that will enable effective utilization of the information.

3.3.2 Security Approaches

3.3.2.1 The Threat Landscape

Although service oriented architecture (SOA) presents a new paradigm for software architecture and design, its implementation shares many features with distributed applications and networks. As SOA ecosystems evolve, two sides of risk emerge: increase through design and implementation vulnerabilities, and decrease by centralization and standardization of controls. Enterprises should plan and implement a strategy centered on a service-oriented approach to traditional controls such as zoning, access control, and encryption. The security team must work with enterprise architects and developers to ensure that risk-appropriate controls are built into the ecosystem at the right stage of maturity.

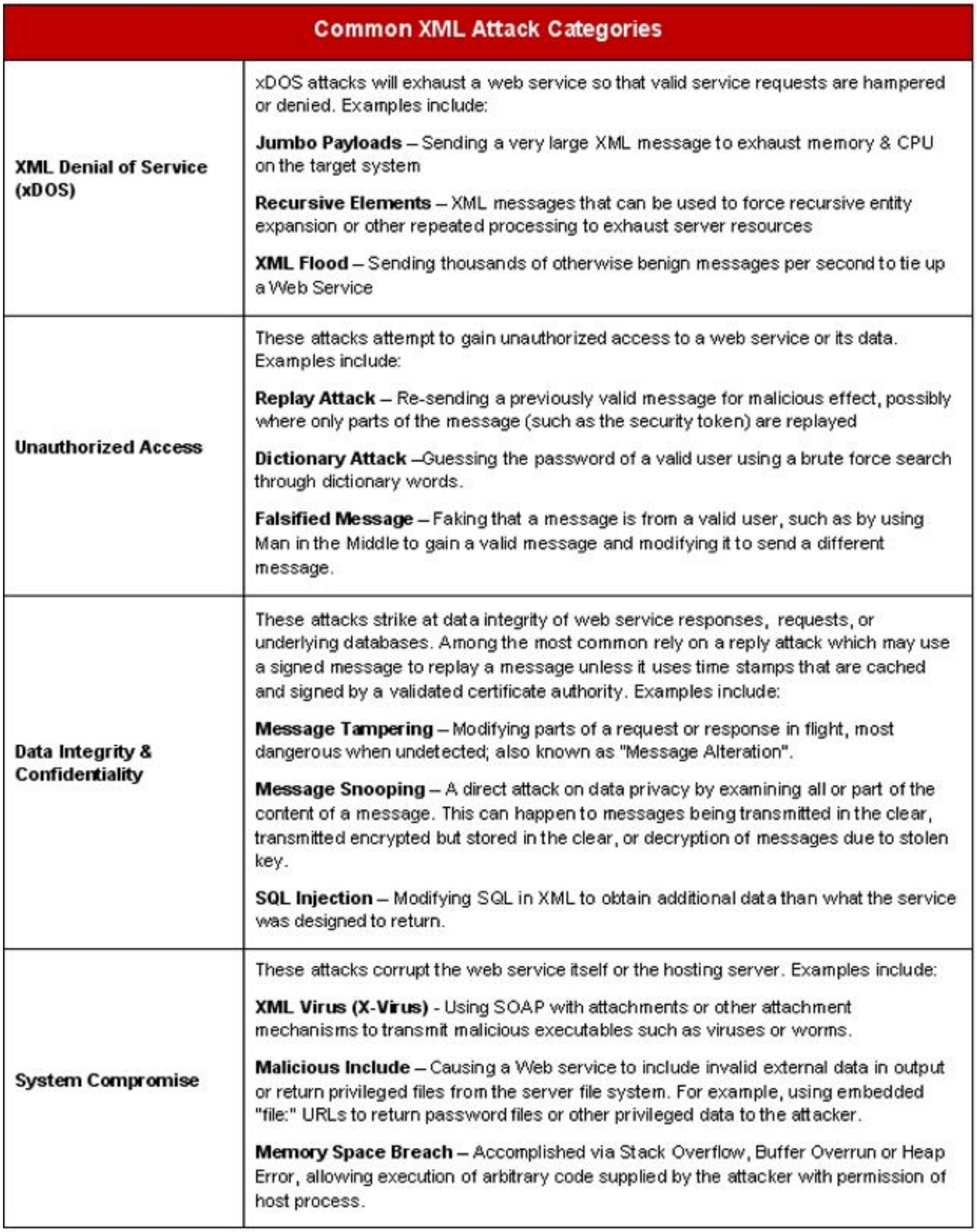
Unlike conventional application architecture, service-oriented solutions that expose Web services to external users and business partners are more susceptible to unauthorized access. Traditionally application architectures allowed limited external access with adequate security controls in place to do so. Message-level security was unnecessary because all processing was carried out internally, within the application. Instead the emphasis of security was on how the message was transported. The contents of the messages were assumed to be valid since they were coming from one trusted source within the organization.

For example, in a service-oriented infrastructure, one solution might invite all HTTP/HTTPS traffic through its firewalls, but most firewalls do little to differentiate between a malicious or valid message. A typical organization’s border and internal firewalls are packet filtering and “allow” all HTTP/HTTPS traffic back to the application server from an uncontrolled zone to a DMZ to a secure zone without inspection. Thus, the service-oriented solution has introduced a hole in the firewall.

The proliferation of Web services has pushed the business case for SOA, for they have enabled unparalleled interoperability among a heterogeneous set of enterprise applications. Along with the increased integration, there has been a paradigm shift on how to secure an application given the new frontier of attacks on interdependent enterprise solutions. The new generation of attacks categorically includes threats based upon XML denial of service, unauthorized access, data integrity, data confidentiality, and system compromise.

Furthermore, SOA-based applications are incubating a new breed of security threats to Web based applications. There are numerous attack scenarios around Web services already being exploited and the list keeps growing. The types of attacks include, but are not limited to: denial of service, replay, message alteration, breach of confidentiality, man in the middle, and spoofing attacks. Close attention to and support for standards such as WS-Trust are vital.

In particular, XML attacks tend to be the most pervasive through well-formed SOAP messages embedded in HTTP/HTTPS protocols. Overall, there are four broad classifications of threats associated with XML, as summarized in the table below.



Common XML Attack Categories	
XML Denial of Service (xDOS)	xDOS attacks will exhaust a web service so that valid service requests are hampered or denied. Examples include: **Jumbo Payloads** – Sending a very large XML message to exhaust memory & CPU on the target system **Recursive Elements** – XML messages that can be used to force recursive entity expansion or other repeated processing to exhaust server resources **XML Flood** – Sending thousands of otherwise benign messages per second to tie up a Web Service
Unauthorized Access	These attacks attempt to gain unauthorized access to a web service or its data. Examples include: **Replay Attack** – Re-sending a previously valid message for malicious effect, possibly where only parts of the message (such as the security token) are replayed **Dictionary Attack** –Guessing the password of a valid user using a brute force search through dictionary words. **Falsified Message** – Faking that a message is from a valid user, such as by using Man in the Middle to gain a valid message and modifying it to send a different message.
Data Integrity & Confidentiality	These attacks strike at data integrity of web service responses, requests, or underlying databases. Among the most common rely on a reply attack which may use a signed message to replay a message unless it uses time stamps that are cached and signed by a validated certificate authority. Examples include: **Message Tampering** – Modifying parts of a request or response in flight, most dangerous when undetected; also known as "Message Alteration". **Message Snooping** – A direct attack on data privacy by examining all or part of the content of a message. This can happen to messages being transmitted in the clear, transmitted encrypted but stored in the clear, or decryption of messages due to stolen key. **SQL Injection** – Modifying SQL in XML to obtain additional data than what the service was designed to return.
System Compromise	These attacks corrupt the web service itself or the hosting server. Examples include: **XML Virus (X-Virus)** - Using SOAP with attachments or other attachment mechanisms to transmit malicious executables such as viruses or worms. **Malicious Include** – Causing a Web service to include invalid external data in output or return privileged files from the server file system. For example, using embedded "file:" URLs to return password files or other privileged data to the attacker. **Memory Space Breach** – Accomplished via Stack Overflow, Buffer Overrun or Heap Error, allowing execution of arbitrary code supplied by the attacker with permission of host process.

Table 3: Common Types of XML Attacks

(Source: "SOA Security 101: Patching the Firewall Hole" The SOA Magazine, Aug 13, 2008)

3.3.2.1.1 Packet Inspection Won't Cut It

Since SOAP messages containing XML payloads traverse across run-time service activities using the HTTP/HTTPS protocols, packet filtering firewalls and routers do not provide adequate security controls. Packet filtering devices will not be able to differentiate authorized well-formed XML messages with malicious messages. However, XML firewalls can bridge this gap.

First, XML firewalls provide packet level inspection of all inbound and outbound traffic to the back-end web application server. For inbound traffic, the device tears open packets to scan its contents against a catalog of threats before it forwards it on to the application for processing. For outbound traffic, the device may apply response filters to inspect packets for sensitive information inadvertently included in the response. Response filters can serve as a critical component to prevent confidential information from leaving an organization through Web services.

Security, integrity and performance characteristics of different applications vary and mismatches can block reuse capabilities.

The proper defense in depth for a SOA environment is:

- Network perimeter defenses
- Identity-based defenses at a centralized entry point
- Identity-based defenses at each intermediary and endpoint
- Security monitoring for attack and fraud detection
- Transport-level and application-level message protections

If users and roles are not well-defined, correct authorization to service-based resources will be compromised. Access policies within the SOA run-time infrastructure depend upon clearly defined entitlements. The risk to the architecture is that ad-hoc solutions to enforce entitlements will quickly become unmanageable.

3.3.2.1.2 Unknown Attacks: The Ugly Details: [25]

We are seeing three primary trends contributing to today's threat landscape:

1. TREND #1 - Building threats is getting easier and quicker. Semi-automated threat generation tools, also known as exploit frameworks, have lowered the bar when it comes to the degree of knowledge and amount of time required to launch ever more sophisticated attacks. This has contributed, at least in part, to the shrinking window of time between when a vulnerability is disclosed to when a specific threat *targeting* that vulnerability is released. Zotob (as well as Witty before it) clearly demonstrated the efficiency of the bad guys, with both worms hitting the information highway less than one week after announcement of the associated vulnerabilities. Indeed, in the first half of 2005, the average time between the disclosure of a vulnerability and the release of an associated threat was only 6.0 days[26]. A related and equally troubling fact is that during this same

period exploit code was published simultaneous with disclosure for 13% of all vulnerabilities2[27].

2. TREND #2 - Threats are spreading across networks more quickly. Threats are increasingly taking advantage of mechanisms that enable them to spread at a frighteningly fast pace. In 2001, Code Red had an infection doubling time of approximately 37 minutes. In 2003, Slammer achieved an initial doubling rate of 8.5 seconds, on its way to infecting 90% of all susceptible hosts within 10 minutes. With the growing ubiquity of high-speed Internet services and evolving capabilities of threat writers, it is really just a matter of time before "flash" threats (requiring just 30 seconds to fully spread) are a reality. When that day arrives, the effectiveness of all countermeasures that are reactive in nature will essentially be reduced to zero.

3. TREND #3 - Threats are getting more elusive. Another byproduct of the previously mentioned exploit frameworks is the capability to rapidly modify existing threats to take advantage of both newly discovered vulnerabilities as well as newly available exploit code. This has led both to a proliferation of threat variants as well as to blended threats being the norm rather than the exception. Furthermore, it is equally troubling that threats are increasingly driving up the computing stack to the application layer. Specifically, they are targeting the plethora of vulnerabilities which exist in applications, and in doing so are making themselves transparent to the predominately lower-layer countermeasures that comprise most organization's defenses. In fact, current estimates are that approximately 80% of all new malware being generated is focused on application-oriented vulnerabilities.

There are several critical consequences and implications of these three trends:

First, the volume of unknown attacks is increasing. Faster threat propagation times means that there is significantly less opportunity to develop and implement updates to countermeasures that operate on the basis of explicitly identifying threats. By definition then, this also means that unknown attacks are becoming more prevalent. Notably, this situation is further compounded by the rapid proliferation of variants, blended threats, and the exponentially growing population of threats that are operating at the application layer (i.e., 'above the radar' of network-layer countermeasures).

A second consequence is that the potential impact of unknown attacks is increasing. As mentioned previously, an unknown attack could be mitigated by having already patched all of the vulnerabilities that an associated threat is attempting to exploit. However, while patching has historically been a very effective countermeasure against unknown attacks, faster threat generation times have caused this to no longer be the case. Today, in most instances a patch is not even available at the time that a new threat is released into the wild. On average patches are being released 54 days after disclosure of an associated vulnerability[28]. This far exceeds the 6-day average for release of a threat2. But even when patches are available—for example, it is not unheard of for Microsoft to release a patch at the same time that it discloses a given vulnerability—the window of time in which they need to be assessed and implemented is simply too short. Most organizations require a minimum of 30 days for routine execution of their patch management process.

3.3.2.2 Vulnerabilities in the SOA Ecosystem

Using the SOA architecture and development philosophy, organizations still face many of the same risks as with traditional distributed applications. The SOA ecosystem creates an additional layer of abstraction on top of existing architecture, and this layer has many of the same properties as traditional networking and distributed computing. As such, the implementation as a whole both inherits vulnerabilities from the underlying components, and introduces additional vulnerabilities from the additions.

A full examination of risks in the underlying technology is out of scope for this overview; however all of the following SOA elements should be reviewed for vulnerabilities:

3.3.2.2.1 Data Format

Although somewhat dependent on the choice of technologies and standards, the formats used for message, description, and policy data introduce vulnerabilities that can cause a wide range of effects in almost all of the components that make up the SOA environment. In addition to the data format, any schema language used to describe the data format may do the same.

3.3.2.2.2 Description

Policy descriptions are used to define and enforce security and other policies in the SOA infrastructure. Built on a common data format, they use standardized policy expressions to ensure interoperability. Although the details are somewhat specific to the exact standards used, certain vulnerabilities can be present in most standards.

3.3.2.2.3 Services

Services are merely abstracted views of their underlying implementation, consisting of recursive layers of logic and infrastructure all the way down to the hardware. A look inside the black box may reveal a top layer of a single software module, a composition of other services, or even the front end to an entire multi-tiered distributed application. As such, even though the implementation details are hidden from the consumer, vulnerabilities inside the black box may be accessible via the service interface, web interface, or an alternate vector on a lower layer.

3.3.2.2.4 Service Platform

Service platforms provide the messaging interface and run-time environment for the services, and may also include part of the managed communication infrastructure.

All of these architecture segments could be subject to any or all of the following:

- Recursive references
- External references
- Unstructured data
- Incorrect specifications
- Inability to specify and/or enforce
- Unauthorized access

- Unauthorized use
- Implementation weaknesses
- Architecture weakness
- Message disclosure, tampering and/or spoofing
- Header tampering and/or spoofing

(Source: "SOA Security: Developing a Technical Control Strategy" Burton Group, Sept. 2008)

3.3.2.3 Building Security In From the Beginning

Good security has to start with secure code. The reason that most attacks can occur is that vulnerabilities exist within the programs themselves. The easiest way to break system security is often to circumvent it rather than defeat it (as is the case with most software vulnerabilities related to insecure coding practices).

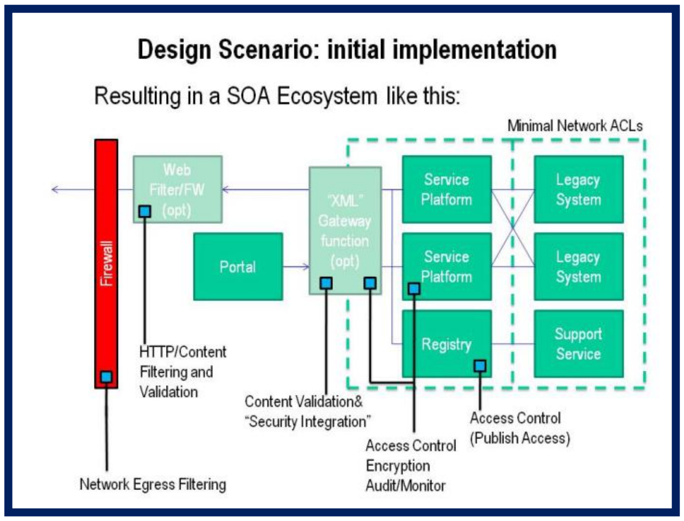
CERT* recommends 12 best practices for secure coding:

1. **Validate input.** Validate input from all un-trusted data sources. Proper input validation can eliminate the vast majority of software vulnerabilities. Be suspicious of most external data sources, including command line arguments, network interfaces, environmental variables, and user controlled files.
2. **Heed compiler warnings.** Compile code using the highest warning level available for your compiler and eliminate warnings by modifying the code.
3. **Architect and design for security policies.** Create a software architecture, and design your software to implement and enforce security policies. For example, if your system requires different privileges at different times, consider dividing the system into distinct intercommunicating subsystems, each with an appropriate privilege set.
4. **Keep it simple.** Keep the design as simple and small as possible. Complex designs increase the likelihood that errors will be made in their implementation, configuration, and use. Additionally, the effort required to achieve an appropriate level of assurance increases dramatically as security mechanisms become more complex.
5. **Default deny.** Base access decisions on permission rather than exclusion. This means that, by default, access is denied and the protection scheme identifies conditions under which access is permitted.
6. **Adhere to the principle of least privilege.** Every process should execute with the least set of privileges necessary to complete the job. Any elevated permission should be held for a minimum time. This approach reduces the opportunities an attacker has to execute arbitrary code with elevated privileges.
7. **Sanitize data sent to other systems.** Sanitize all data passed to complex subsystems such as command shells, relational databases, and commercial off-the-shelf (COTS) components. Attackers may be able to invoke unused functionality in these components through the use of SQL, command, or other injection attacks. This is not necessarily an input validation problem because the complex subsystem being invoked does not understand the context in which the call is made. Because the calling process understands the context, it is responsible for sanitizing the data before invoking the subsystem.
8. **Practice defense in depth.** Manage risk with multiple defensive strategies, so that if one layer of defense turns out to be inadequate, another layer of defense can prevent a

security flaw from becoming an exploitable vulnerability and/or limit the consequences of a successful exploit. For example, combining secure programming techniques with secure runtime environments should reduce the likelihood that vulnerabilities remaining in the code at deployment time can be exploited in the operational environment.
9. **Use effective quality assurance techniques.** Good quality assurance techniques can be effective in identifying and eliminating vulnerabilities. Penetration testing, fuzz testing, and source code audits should all be incorporated as part of an effective quality assurance program. Independent security reviews can lead to more secure systems. External reviewers bring an independent perspective; for example, in identifying and correcting invalid assumptions.
10. **Adopt a secure coding standard.** Develop and/or apply a secure coding standard for your target development language and platform.
11. **Define security requirements.** Identify and document security requirements early in the development life cycle and make sure that subsequent development artifacts are evaluated for compliance with those requirements. When security requirements are not defined, the security of the resulting system cannot be effectively evaluated.
12. **Model threats.** Use threat modeling to anticipate the threats to which the software will be subjected. Threat modeling involves identifying key assets, decomposing the application, identifying and categorizing the threats to each asset or component, rating the threats based on a risk ranking, and then developing threat mitigation strategies that are implemented in designs, code, and test cases.

(*CERT is an organization devoted to ensuring that appropriate technology and systems management practices are used to resist attacks on networked systems and to limiting damage and ensure continuity of critical services in spite of successful attacks, accidents, or failures. Source: https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices;jsessionid=4DB87F4B348076759D2FDEE8E85FEB59)

3.3.2.4 Security Depends on the Implementation


Design Scenario: initial implementation
Resulting in a SOA Ecosystem like this:
Minimal Network ACLs
Firewall
Web Filter/FW (opt)
Portal
"XML" Gateway function (opt)
Service Platform
Service Platform
Registry
Legacy System
Legacy System
Support Service
HTTP/Content Filtering and Validation
Content Validation& "Security Integration"
Access Control Encryption Audit/Monitor
Access Control (Publish Access)
Network Egress Filtering

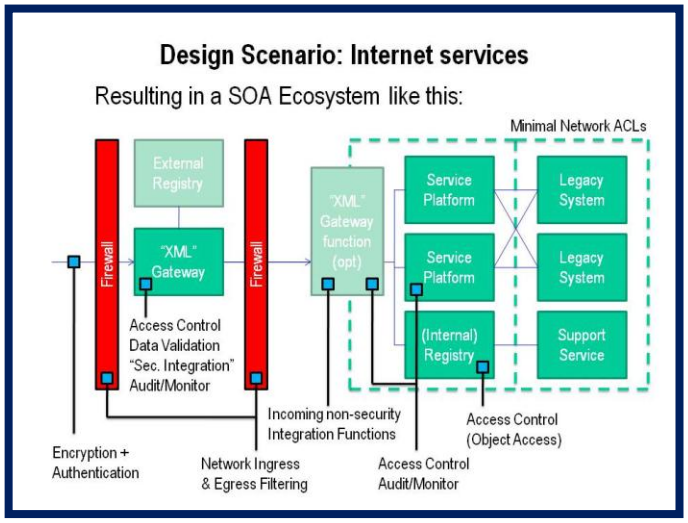
Depending on the use of the SOA project, security surrounding the implementation can go from minimal to mission critical. Below are three examples of how the security measures can differ. The first is an initial, minimal implementation. It includes the following technical controls:

- Transport-level security
  - The FAA's managed telecommunication infrastructure (FTI) and back-end systems access restrictions
  - Egress filtering and encryption for outgoing service connections
  - Encryption and authentication for service connections (optional)

- Message-level security
    - Access control at the FTI perimeter (optional)
    - Content validation at the FTI perimeter (optional)
    - Egress filtering for outgoing service connections (optional)
    - Registry Security
    - Access control for publish operations, optionally for read operations
    - Auditing and monitoring

This basic level of security ensures that bad connections and content is eliminated and that only authorized individuals can access and/or publish to services.

The second example is when services are exposed to the Internet. This increases the risks from:
- Unauthorized service access
- Message confidentiality and integrity
- Malicious data threats to all components
- Registry security for external services
- Security protocol incompatibility

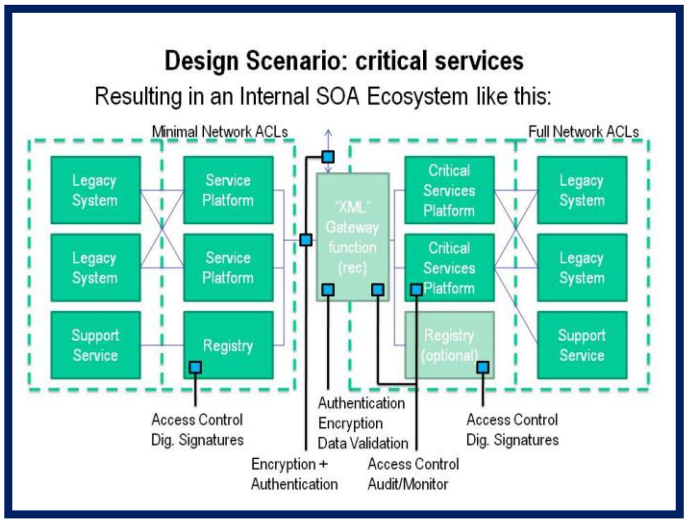

Design Scenario: Internet services
Resulting in a SOA Ecosystem like this:
Minimal Network ACLs
External Registry
"XML" Gateway
Firewall
Firewall
"XML" Gateway function (opt)
Service Platform
Service Platform
(Internal) Registry
Legacy System
Legacy System
Support Service
Access Control
Data Validation
"Sec. Integration"
Audit/Monitor
Incoming non-security Integration Functions
Access Control (Object Access)
Encryption + Authentication
Network Ingress & Egress Filtering
Access Control Audit/Monitor

To compensate, one needs to add the following technical controls:
- Transport-level (network) security
    - Ingress filtering for incoming service connections
- Message-level Security
    - Demilitarized Zone XML gateway
    - FTI perimeter XML gateway function (optional)
- Transport-level or Message-level Security
    - Encrypted channel for incoming service connections
    - Authentication for incoming service connections
- Registry Security
    - Separate registry – either by full separation or object access control
- Auditing and monitoring

Contrast the two previous examples with the security needs for mission-critical applications. In this scenario there are increased risks from:
- Unauthorized service access
- Unauthorized service and infrastructure changes
- Service availability and performance
- Message confidentiality and integrity
- Malicious data threats
- Registry security
-

To compensate, one needs to add the following technical controls:
- Transport-level (network) filtering
    - Strict network separation of critical services and infrastructure

- Transport-level or message-level security
    - Encrypted channel for connections to critical services
    - Authentication for connections to critical services
- Registry security
    - Separate registry – either by full separation or object access control
    - Cryptographic signing of registry entries
- Auditing and monitoring
    - Critical service platforms
    - Other critical FTI components (e.g. XML gateway)


Design Scenario: critical services
Resulting in an Internal SOA Ecosystem like this:
Minimal Network ACLs
Full Network ACLs
Legacy System
Legacy System
Support Service
Service Platform
Service Platform
Registry
"XML" Gateway function (rec)
Critical Services Platform
Critical Services Platform
Registry (optional)
Legacy System
Legacy System
Support Service
Access Control Dig. Signatures
Authentication Encryption Data Validation
Encryption + Authentication
Access Control Audit/Monitor
Access Control Dig. Signatures

This increased level of security provides for full digital signatures, encryption, as well as authentication added to the above scenario.

(Source: "SOA Security: Control Architecture Design Scenarios" Burton Group, Sept. 2008)

3.3.2.5 Security Technologies to Mitigate Risk

There are several critical technologies involved in mitigating risks to a SOA environment. The first is being able to manage users: know who they are, what they are allowed to do, and monitoring their compliance.

The second is to manage the content, ensuring that bad or unwanted content is kept out and that good content is delivered, with high levels of integrity, where it belongs … and nowhere else.

3.3.2.5.1 Identity and Access Mgmt.

Access control is one of the most important aspects to consider in a SOA environment. Who can get where and do what when they get there is an extremely complex issue.

3.3.2.5.2 User Identity Gets Lost in Translation

Enterprises have clearly created mechanisms and adopted technologies that open up enterprise networks and business applications to the outside world. The level of access control to these networks that was considered adequate within the confines of the office buildings is no longer acceptable. The prospect of a hacker gaining access to enterprise systems claiming to be an employee is enough to make even the most experienced IT administrators queasy.

Moreover, the concepts of managing user identity, in most cases the primary element to controlling the "keys to the kingdom" have been relatively overlooked, much to the detriment of unsuspecting organizations. Most companies have implemented strong security measures to protect physical access to their facilities. And access to the network from inside the perimeter is often seen as a priority. User authentication with passwords has served as a means of establishing user identities since the beginning. The system worked fine when the community accessing information did so when inside the secured physical boundary of the organization.

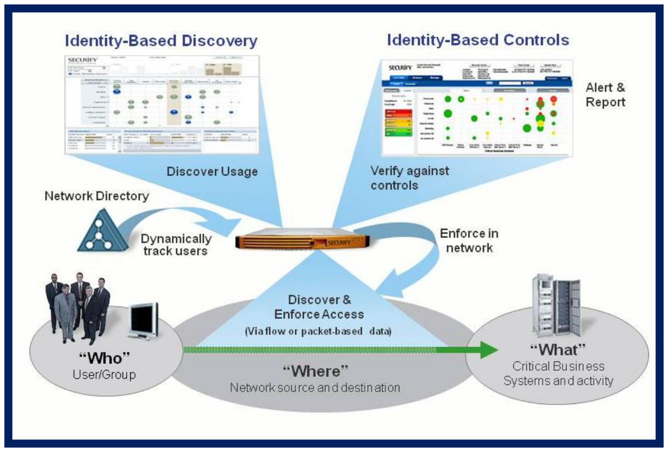
Over the years, increasingly sensitive data on networks has been exposed to more users, attracting more hackers to break into enterprise network environments. Access by a large user community leaves enterprises even more vulnerable to hackers since security of the system is now dependent on the strength of the weakest password in a large group of passwords.

However, relying on simple usernames and passwords to access the entrances to these secure gateways is similar to leaving the front door to your house unlocked, or using a lock that can be easily picked. Static passwords are an unreliable mechanism for guarding the entrance to your trusted systems, applications, and networks. Many passwords can be easily guessed, hacked, or compromised by brute force attacks.

Even complex password policies present problems for end users and IT departments. Changing passwords every 30 days, not allowing users to repeat a password over a given time period and requiring multiple special characters in passwords adds significant complexity. Users may simply forget their password, requiring a call to the help desk to reset the password and driving up the overall cost of IT support and lost productivity, to say nothing of diminished user convenience.

The vulnerabilities of static passwords as an identity mechanism have been well-documented, raising the need for stronger methods of user authentication. Companies need strong authentication solutions that not only provide reliable security, but that are also easy to install and deploy, simple to manage, and able to grow with their needs.

Strong authentication—also known as two-factor authentication—refers to systems that require multiple factors for authentication and use advanced technology, such as secret keys and encryption, to verify a user's identity. The simplest example of strong authentication is your ATM card. This requires something you have (your card), and something you know (your PIN). Most people wouldn't want their bank to allow access to their checking account with just one factor. Yet many organizations allow entrance to their valuable resources (often much more valuable than a single personal checking account) with only one factor—a weak password!



Two factor authentication solutions deliver security through one-time passcode-generating hardware tokens combined with the user's PIN. The user simply pushes the button on the token, generating a single-use passcode (via a unique secret key and an advanced encryption algorithm that is contained inside). The user enters the passcode, followed by the user's unique PIN, to gain access. After one use, the passcode is thrown away by the system. If someone attempts to re-use a passcode, access is denied by the authentication server.

3.3.2.5.3 Where Are They Going? What Are They Doing?

In addition, once the user is granted access it is essential to monitor access policy in real time. True security relies on the ability to verify access and usage, track compliance, and alert on policy violations in real time.

3.3.2.5.4 Content Management

As we discussed above, security relies on the ability to control both the users and the content. The next few sections will go into more detail about protecting content as it moves in and out of the network.

There are two major ways to control networks: keeping out the “known” bad (negative security) or only allowing in the “known” good (positive security). Let’s discuss the pros and cons of both.

- **Negative-Model Countermeasures**

  Negative-Model countermeasures are countermeasures based on identifying those bits of traffic and communication sessions which are known to be bad, presumably because they are associated with a threat. Antivirus and intrusion detection/prevention systems are classic examples of this type of countermeasure. The old technology of rules and signatures is no longer enough with new threats combining many attack vectors at once and coming from outside and inside the network, and even through encrypted protocols.

- **Positive Security Models**

  Positive-Model Countermeasures are countermeasures based on identifying those bits of traffic and communication sessions which are known to be good, and then allowing only them to proceed, thereby excluding all other traffic and communications. Router ACLs (access control lists) and firewalls are classic examples of this type of countermeasure. They should be configured to expect specific types of traffic or protocols that are implemented and guard against all that is not.

Much has been made of the importance of a firewall’s *policy*—the rules by which you instruct a firewall what to permit or deny—but the firewall’s policy is only approximately half of the story. To use an analogy, the firewall is a guard at the perimeter of your network which you have instructed whom to shoot and whom to allow past. Obviously, it’s important that your guard remembers the rules you gave him (the policy) but it’s equally important that your guard uses reliable means to determine if the people he’s letting back and forth are truly who they appear to be, and in addition that they are not carrying anything dangerous in their pockets [29]

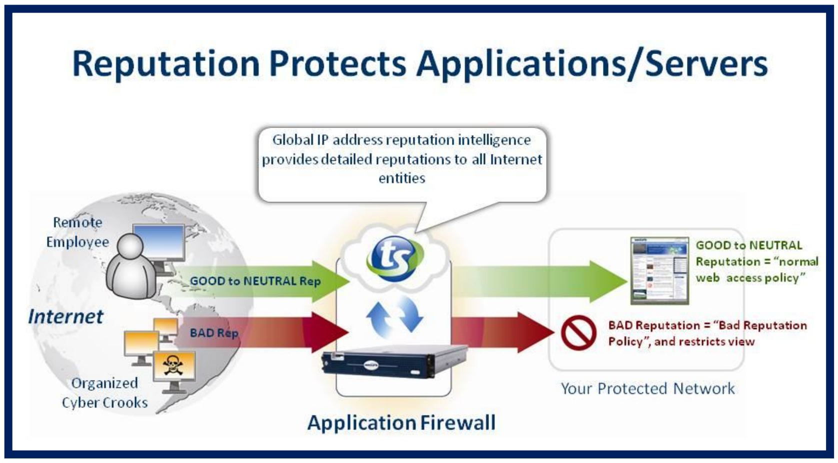
3.3.2.5.5 Reactive Systems Are Not Enough

Unfortunately, many enterprises rely on an e-mail security solution based solely on message content and identifying virus signatures. However, the exponential growth in e-mail volume means that substantial hardware and bandwidth is required in order to analyze every message for spam and viruses.

In addition, signature engines are unable to keep up with outbreaks caused by zombie networks, which are responsible for the majority of spam and viruses being sent today. These networks of hijacked computers send millions of messages a day, with thousands of new zombie machines being created daily. As a result, corporate networks are being hit with a constant stream of unwanted messages; these messages must be handled before they can flood the mail server.

Today's viruses propagate at startling speeds. Consider, for example, the "Code Red v2" worm, which infected 400,000 hosts in about 14 hours. While this window of time allowed the leading anti-virus vendors to release signatures to combat the worm, "Code Red" was only a precursor to the attacks seen more recently. The "Warhol" worm needed only 15 minutes to infect its first 1,000,000 hosts, and the more recent "Flash" worm infected the same number in just 30 seconds. With propagation speeds like this, one can easily see why relying only on signature-based virus detection is no longer a viable option.

3.3.2.5.6 Reputation-Based Behavior

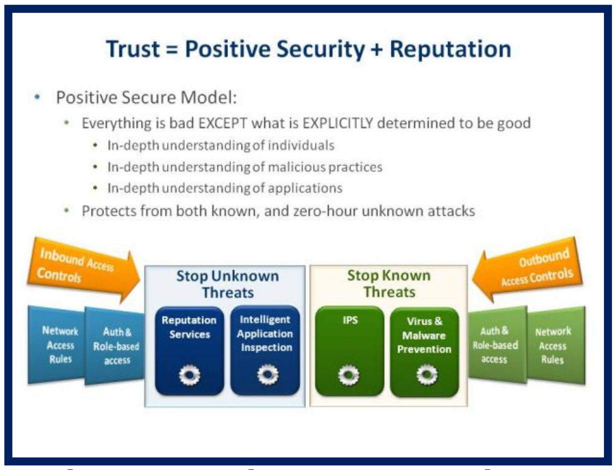

Reputation Protects Applications/Servers
Global IP address reputation intelligence provides detailed reputations to all Internet entities
Remote Employee
GOOD to NEUTRAL Rep
Internet
BAD Rep
Organized Cyber Crooks
Application Firewall
GOOD to NEUTRAL Reputation = "normal web access policy"
BAD Reputation = "Bad Reputation Policy", and restricts view
Your Protected Network

One of the newest trends in security is to analyze behavior and provide a reputation score, similar to a credit score in the real world. Quite simply, a reputation system is designed to track the behavior of network entities such as IP addresses, domains, URLs, images, and messages. Some trends that are evaluated to form a reputation include whether a sender or host engages in good behavior (such as sending legitimate email messages or hosting a malware-free Web site), bad behavior (such as sending spam or malicious code) or is deviating from known historical behavior. Also incorporated in a reputation is affiliation; for example, has a given IP address hosted malicious URLs in the past?

Adding reputation allows administrators to adds an extra layer of proactive protection to these solutions by analyzing and correlating the following elements:

- **Volume of reputation data.** The more information that can be viewed and analyzed, the better the results.
- **Quality of reputation data.** Because of the basic differences between consumer and business usage, the right reputation service should depend primarily on business and government volumes.

- **Accuracy of reputation data.** The ability to perform real-time behavior analysis using over dozens behavior classifiers that examine 1000's of characteristics provides the best and most accurate reputations
- **Combination of sources.** The ability to combine information from multiple sources including virus detection, Web pages, and email traffic enables a reputation service to gather more data than any single source and to aggregate intelligence to help identify and stop blended threats that use multiple protocols.
- **Strength of multiples.** The right reputation service should be able to combine sender, message, domain, image, and URL reputations and correlating the relationships.


Trust = Positive Security + Reputation

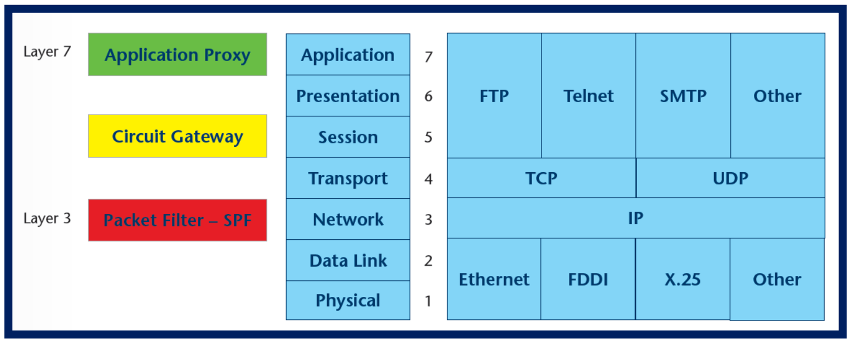
- Positive Secure Model:
  - Everything is bad EXCEPT what is EXPLICITLY determined to be good
    - In-depth understanding of individuals
    - In-depth understanding of malicious practices
    - In-depth understanding of applications
  - Protects from both known, and zero-hour unknown attacks

Adding reputation to the positive security model creates what we call a "trusted" security model … the highest level of protection against both known and unknown attacks and attackers.

3.3.2.5.7 Application Layer Firewalls

We already discussed above that just packet filtering is insufficient for true security.

All firewalls rely on the inspection of information generated by protocols that function at various layers of the OSI model as shown in the Figure below. Knowing the OSI layers at which a firewall is capable of operating is one of the keys to understanding the different types of firewall architectures.



Generally speaking, firewalls follow two known rules:

- The higher up the OSI layer the architecture goes to examine the information within the packet, the more processor cycles the architecture applies to examining traffic flows
- The higher up in the OSI layer at which an architecture examines packets, the greater the level of protection the architecture provides, since more information is available upon which to base security decisions

An application firewall (also called application filtering firewall) limits the access which software applications have to the operating system services, and consequently to the internal hardware resources found in a computer, much as a car firewall limits access of heat, or even fire, to the passengers of the vehicle. The reason that application firewalls are needed in today's internet and data-sharing world is that the other types of firewalls in existence do not control the execution of data, only of the flow of data to the computer's processor.

Strong security requires firewalls that are application-aware and can control access and content with fine granularity.

- Minimal and predictable software maintenance
- Rugged and virtual appliances to fit into any environment
- Integrated high performance
- Single appliance to stop both known and unknown attacks
- Eliminating unauthorized application access and usage by enforcing the "trusted" security policy model
- Proxies - removes hacker visibility to your applications and networks
- Complete bi-directional content inspection of encrypted traffic
- Visibility and policy enforcement of encrypted application sessions
- Blocks access attempts by and to untrustworthy parties
- Removes inbound and outbound exposure to risky geographies, shrinks the Internet
- Protection against new attack variants
- Geo-Location aware so that only connections from approved countries are granted access

3.3.3 Interaction Patterns and Orchestration

The enterprise integration pattern development work published in *Hohpe, Gregor and Woolf, Bobby*. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. *Addison-Wesley, 2003*, particularly the work of Hohpe and Woolf, is the basis for much of the assumptions in this section. While their work centers on messaging solutions, it is their understanding and appreciation for large-scale, distributed systems in general that must be appreciated above and beyond the basic patterns they describe. The nature of the SWIM program makes Hohpe and Woolf's descriptions of tradeoffs very relevant to large SOA deployment scenarios. With SWIM, there are technical and organizational challenges, largely associated with differing levels of technology adoption, across the Air Traffic Organization as a whole.

In particular, their descriptions of loose coupling are the most relevant to SWIM. As described in the section on Message Oriented Middleware, and because systems in the ATO are, by definition, based on legacy requirements, loose coupling is a necessity. The legacy systems may not have been built with loose-coupling, or asynchronous messaging in mind, but they must continue to service the needs of important elements of the nation's aviation infrastructure. Aviation itself is changing at a very fast pace, and the technology used by aviation is a major contributing factor to that evolution. In order for the ATO to keep abreast of advances, changes need to be made to the infrastructure while the systems continue to operate in a reliable and predictable fashion. Loose coupling concepts, along with the major tenets of large-scale system interaction covered so ably by Hohpe and Woolf, can provide useful guidance to system integration efforts supported by SWIM.

SOA can be accomplished through a variety of interaction patterns. Our recommendations on the Shared Service approach should be augmented by a detailed study of the potential for large-scale, asynchronous messaging to play a key role in the nature of the interaction patterns between and among SIPs.

3.3.4 Best Practices in XML document design

XML data representation is often used in SOA implementations. XML representation is a mature technology with well defined standards and supporting tools. Following are a brief set of points to consider regarding XML representation.

- XML is one representation of many available for use in SOA implementations. It should not be considered the only representation available and should only be used when appropriate. See Section 2.3.1 for more information.
- XML validation, transformation and authentication can be processing intensive. XML specific hardware devices can be used to accelerate this processing.
- There is an extensive body of knowledge around this mature technology. Additional information can be found through the broad based World Wide Web Consortium (W3C):

http://www.w3.org/XML

3.3.5 System Engineering for SOAs

System Engineering (SE) provides a framework for organizing and managing complex undertakings such as SWIM. The extensive body of knowledge of SE applies to development of a SOA, with a few adaptations learned over the past several years to accommodate service-orientation. This is an important topic because the consequences of a narrow approach to SWIM development may be that the FAA will not realize the full value of the investment in energy and money. Not following a disciplined SE process is not a showstopper, but it increases cost and risk at every phase. Decisions that shortcut a disciplined SE approach have proven repeatedly to be costly.

In describing an SE approach for SWIM, this section focuses on the following three areas:

1. This section focuses narrowly on special SE considerations for a SOA based on heuristics, and assumes that the reader is somewhat familiar with the tools and principles of SE.
2. This section builds on the GEIA white papers on “SOA Best Practices” and “SOA Governance” by putting some of the key recommendations from those papers in the context of a disciplined SE approach.
3. In keeping with the overall theme of this paper, this section addresses the subject of special SE considerations for planned transitions between program Segments.

3.3.5.1 Foundational Principles

3.3.5.1.1 The Fundamental Role of SE Remains the Same

SE is a multi-disciplinary practice that takes a holistic approach to system development. It is a systematic approach, especially useful for planning, organizing and executing complex systems. The SE practice encompasses the specialties of system architecture, system analysis, operations research, decision analysis, requirements development, risk management, process engineering, and others. This paper uses the generic term "SE" to describe any or all of the specialties within SE; and because the focus of this paper is on SOA, much of the SE section is about the role of the system architect.

The specialized role of the system architect is to design the system. The architect balances cost, performance and schedule within the constraints of the environment, which include technical, political, programmatic, and other constraints. For SWIM, these factors determine the design trades among federated, consolidated, and shared services alternatives. The FAA's choice of a service-oriented style of architecture for SWIM has unique implications to all levels of the system, but the fundamental role of the system architect, and in general, SE, remains unchanged because of this architecture decision. In other words, the principles of SE apply equally as well to a SOA as to other styles of architecture.

3.3.5.1.2 The SE Perspective

The unique SE perspective described here helps explain some the fundamental principles of SOA design and SE priorities for system development found in the following major section on the "SE Approach for SOA."

One of the reasons that SE principles are so widely applicable is because SE looks at generic "components" (nouns), functions (verbs) and relationships (a.k.a. interactions, interconnections) among these objects in a complex system. And the greater payoff in SE results from a focus on what the system *does* (functions) vs. what it *is* (components). This emphasis balances the functional groups more typically concerned with design of the components.

SE also transcends various taxonomy that describe the relationships among parts of the system. Though rival descriptions of a system are vigorously debated, one's view of the system depends on where one stands. Examples can be found in the finer descriptions of enterprise architecture, solutions architecture, reference architectures, architecture frameworks, object-orientation, service-orientation, layers of the stack, global, local and more. SE will translate, allocate, and relate. In SE, the inherent value of different abstract views of a system is determined by their practical usefulness in managing overall system development to meet the user's mission objective. SE is big-thinking - having the whole picture in mind, even if the picture comprises many different simultaneous views.

SE embraces the lifecycle concept described in the SOA Governance white paper. The role of SE is "keeper of the vision" through every transition and throughout the entire lifecycle of the system, and as such, contributes much to the governance structure. For building the system, there are several models of system development, such as the classic V-Model and newer Agile development. The FAA ATO has adopted its own model defined in the NAS SE Manual. Basically, an SE model describes business and organizational requirements and how services are developed, integrated, and managed throughout their lifecycle. Some common attributes of these models important to SE are top-down, bottom-up, and iterative processes incorporating feedback (for interdependencies and throttling) at every step in development.

The aim of SOA is to de-couple the services from underlying technologies and move toward the paradigm of "diverse nouns and fixed verbs" pattern for interoperability. However, the fact that services and nodes in the system are not equal in practice causes the need for situational application of supporting components. The organizational models used for managing this complexity such as by "services" or "capabilities" or others in real complex systems have been proven useful, but incomplete. The SE must use dozens of models simultaneously, and deploy them situationally.

3.3.5.1.3 Distinguishing Characteristics of SOA

The distinguishing characteristics of a SOA are:

- Chunks of functionality in the form of services (software) that can be used many times and strung together in many combinations to build new applications (a.k.a. modularity and composability)
- Relative independence (a.k.a. loose coupling) among functions, orchestration, transportation mechanisms, technologies, and data, which enables abstraction, re-use, interoperability, and the unique relationship between provider and consumer

3.3.5.1.4 Benefits of SOA

As described in Section 1 of this paper, a simple architecture of re-usable, independent services can have many benefits, such as:

- Cost savings from re-use of relatively independent new and legacy services, sharing of services across the enterprise, and decreasing marginal cost of building new applications.
- Better quality information that is timely (real-time or nearly so) and widely shared across the enterprise by an owner who controls the master source at the endpoint.
- Easier integration of new services that enables agile (and faster speed) business transformation for a complex organization, unencumbered by IT dependencies.

Recalling the specialized role of the system architect, these benefits again highlight the parallel responsibility of SE for a SOA in balancing cost, performance and schedule in the design of the system.

3.3.5.1.5 How the Benefits of SOA are Achieved / Elements of Good SOA Design

To help achieve the promised benefits of SOA, some of the following implementation techniques are being employed:

- An approach to SOA design as a business enterprise function vs. an application or technology approach puts the focus on the information required for critical points in the decision-making process and the creation of business value.
- Positioning the architecture in the appropriate environment helps achieve the "ilities" (reliability, maintainability, survivability, flexibility, etc).
- The SOA may make use of a common service bus or service container implemented locally, globally, or a hybrid configuration to provide autonomous overhead functions in an independent implementation.
- Services are exposed, described and discoverable, which encourages re-use.
- Open source protocols and standards facilitate interoperability, and flexibility by creating loose coupling with the transportation mechanism. Full knowledge of the interface ideally allows a service to be used as a functional black box.

3.3.5.2 SE Approach to SOA

This section describes a few key SE issues for SOA. More comprehensive resources on SE best practices include the FAA NAS System Engineering Manual (SEM) and others.

3.3.5.2.1 Design Principles for the Basic Unit of SOA: a Service

The basic building block of a SOA is a service. It is a unit of software code that performs a particular function. Proper design of services helps realize the potential benefits of SOA.

A service provided by software code is *analogous to a service provided by a person*, such as a cashier, barista or engineer. Extending this analogy, services arranged in a SOA are like *people in an organizational structure* (also an architecture). Some people provide more specialized services than others. A single person's general services can be applied to a great number of different problems. This is analogous to the wide applicability of generalized SOA Core Service. A SOA seeks to gain a similar advantage to teams of different labor specialties, that, when pooled together in various combinations, can be applied to an infinite number of problems. This is analogous to SOA's ad hoc arrangement of services into a great variety of tailored applications.

Migration of IT systems toward a SOA is following the evolution of modern organizational structures in business, government and military that make use of the adaptability of people to different situations. In the 1980s, business markets were slower and more predictable, and the world was in the midst of the Cold War. *Specialized systems* for business and battle prevail until the environment changes. In battle, specialization is an advantage as long as there is control of the geographical context, as there are business advantages to specialization within predictable regulatory environments.

In contrast, SOA trades away some efficiency in architectural specialization to gain speed and adaptability. In response to today's less predictable environment, people are often layered in ad hoc teaming arrangements (such as matrices, red teams, COIs, etc.) to address a particular problem. These often include people from different affiliations, as in a SOA, the service can be provided by any producer (an end point). A person can also reach across organizational boundaries to bring together a common business function. One's organizational affiliations are much looser, as evidenced by the many bosses we have. The goal of SOA is for similar sharing of services used to provide enterprise-wide functionality.

A person's service is flexible, mobile, and can be shared among many tasks and project teams. When a new project or campaign is started, people can be quickly re-arranged into a different configuration, filling in with some new skilled labor or learning new skills, as necessary. This is analogous to orchestration in a SOA to build new applications from a mix of services from the global pool, which allows greater flexibility and uniformity. A much more difficult and time-consuming alternative would be to find a brilliant individual with all the requisite skills to solve a particular complex problem, through either recruiting or building the skills from scratch through education and training. This is more familiar to traditional IT systems optimized for a particular business model.

Services are more abstract than systems, but are conceptually similar in SE. The service is an abstraction of a function, broken down to distinct units. These distinct units can be treated similar to a component, element, or subsystem in a system architecture. Culturally, tangible products are easier to value and manage than intangible services. People are wired that way through their cultural frames and metaphors. For example, domestic economic output, based on industrial age models, measure the contribution of the iPod to the balance of trade by the value of the physical device imported from China, and count Apple's R&D as a business expense. In this model, the iPod contributes to the U.S. trade deficit. Yet even a casual observer of Apple's parking lot recognizes that the real value the iPod to the U.S. economy is much more than value of plastic and transistors. Although the value of the architecture and creativity is not measured, these intangibles are the key to the iPod's economic success. Fortunately, SE has already addressed this issue. New conceptual models for service-orientation have been incorporated into existing SE models (DoDAF, SysML, capabilities-based acquisition) and are in use by the community today.

3.3.5.2.2 Design Principle of Independence

Independence (a.k.a. "loose coupling") is a desirable quality in system architecture for many reasons, including adaptability, scalability, and clean interfaces. Independence can also empower innovation within a SOA environment, just as in organizations.

Independence is achieved by:

- Independent services run inside wrappers such as Java or .NET that allow ad hoc and late binding, and provide some degree of indeterminate data typing.
- Services are appropriately-sized (and naturally-sized) units of functionality (a.k.a. chunks of software code), as described in more detail below.
- Services are intrinsically unassociated with other services, and do not have calls to each other embedded in them. In other words, the services are instantiated as a group of peers rather than as a class hierarchy.
- Service providers are loosely coupled with consumers.
- Ownership of services is at a lower level (which also facilitates a simpler method of managing a complex system, complex because of large scale or broad scope).
- Business models, capabilities and functions are separate from the underlying technologies, though achieving this ideal is elusive in practice.

- Information is decoupled from transportation mechanisms.

3.3.5.2.3 Design Principle of Natural Scale of Chunks / Granularity

The basic building block of a SOA, the service, is an appropriately-sized unit of software code that performs a particular function. The goal of SE is a natural level of specialization of functions into independent services that can be shared and re-used. The right level of "granularity" or "chunkiness" is subjective, and varies locally. Recalling the analogy of teams of people organized for a task, the naturally smallest level of division could be a single laborer, or it could be a group of people acting as a unit, such as an infantry squad of 12 soldiers. Regardless of size, the services should have consistent policy implementation.

A trade study should be performed on this issue to determine the level of granularity that is appropriate for SWIM Segment 2 services. The tradeoffs in granularity are:

- Small and abstract enough to be applicable for re-use
- Enough functionality to be recognizably useful to the user and map easily to business functions that affect users directly
- Large enough to minimize the number of interfaces required to implement an application, which keeps processing overhead low, and meets performance requirements
- Large enough to create an application almost entirely from an orchestration of the service components

3.3.5.2.4 Importance of Boundaries and the Glue Between Services

Boundaries and interfaces are the greatest point of leverage in SE, and their relative importance may be even greater in SOA. The consumer cannot be expected to know the detailed design of the service, so the behavior of the service is defined by the interface. Likewise, the provider can't make many assumptions about the behavior of the consumer.

In SE for SOA, the trick is the glue between services. Services by themselves do not make a functional SOA, and the greatest value comes from the principle that "the whole is greater than the sum of the parts." To realize this benefit, it takes a net-centric approach and glue between services. In migrating from legacy systems to SOA and from SWIM Segment 1 to Segment 2, the glue between services is where a significant number of design and implementation choices are made. Here is where the rubber meets the road, and the greatest impact of higher-level decisions about architecture and migration models is felt. And here again, is where the art of SE comes in.

People and middleware are essential glue in building a SOA. First, a business process expert is needed to orchestrate the services to meet a new or existing business system requirement. Next, the SOA is knitted together with middleware that enables services to work with one another. Some examples are messaging between services, IT infrastructure, bridges to abstract protocols, adapters that plug in legacy systems, data federators, transport mechanisms, and components to reduce interaction latency and enforce security. This step is often trivialized and over-simplified by neat representation in architecture diagrams and solutions marketed as drag-and-drop capabilities. While these elements may not be technically challenging in themselves, implementation often presents unanticipated challenges. SOA is not the collection of technologies that enables interoperability, rather it is more about the application of these technologies.

3.3.5.2.5 Approach as a Top-Down, People-Driven Process

SE begins as a *top-down process*. SOA implementation is optimized by starting with the business processes, rather than a smaller set of objectives such as specific implementations, projects, or programs. This is why SOA can be challenging to an organization, and take real commitment by leadership, not only the IT group within an organization.

SE is also a *people-driven process*, a necessary element of success for a SOA implementation. SE manages complexity in areas beyond the technical issues. Following a rigorous SE process by itself does not guarantee success, and may result in a system that is not fit for the user, discovered during validation phase. And SE is something that needs to be integrated into the culture of an organization. If it is not already there, then SOA will feel more like a cultural change, too. The technologies for SOA are readily available, but it still takes people and strong leadership to pull it off. People manage change in context with other systems and the environment. And experienced people help avoid mistakes and save money.

Given that SE is a top-down, people-driven process, it is not surprising that the *organizational approach for a successful SOA* is also top-down and people-driven. Because SOA often crosses internal and external lines of business and may even require a change in culture, SOA development requires enterprise-level management. Examples of SOA initiatives in government where enterprise-level management were implemented include the TSA Operational Application Support and Information Services (OASIS) and the DoD Intelligence and Information System (DoDIIS). In each case, the challenges of SOA were addressed through leadership of the initiative from the top of the organization. While the specific architecture design is not independent of the domain, common heuristics apply.

Looking at another more generic example, the positions of Chief System Architect (CSA), Chief System Engineer (CSE), and Chief Information Officer (CIO) are close to the top of a program or organization for similar reasons that SOA management belongs near the top of the organization. The paper on SOA Governance states: “SOA provides a distinctive enterprise-level approach for designing and delivering cross functional initiatives, closely involving both business and IT in the collective pursuit of the enterprise’s strategy and goals.” This reads like part of the job description of a CIO. The SOA Governance paper also mentions ownership and funding that is best executed by a top-down organizational construct. The CIO approach centralizes administrative functions and allows operational departments to focus on their mission-critical responsibilities. Managing to SOA requires strong oversight, because growth through satisfying individual interests as they arise is not necessarily the most efficient use of resources.

The CSE/CSA must know everything that is going on. The information that is needed to manage a SOA is not found in any single source. It comes from discussions taking place at all levels with stakeholders, informal emails, regulations, and industry presentations. Some documents and information have precedence over others, and it takes a person with a top-level view who can look at the problem in its entirety to sort it out. This person has to keep the design and key drivers in mind to mentally do the verification and prioritize integration and interoperability issues that need to be overcome. The CSE/CSA fill gaps in guidance and authority derived from the traditional decomposition of systems-level documentation and "policies." Here is another reason why SE is a people-driven process and why the top-down guidance and authority for the SOA must be provided by people, and not by policies alone.

SE, through a top-down and centralized approach, helps accomplish the goal of SOA to unify how business is accomplished across traditional organizational silos. Having the whole picture, SE can consistently apply standards and process, control configuration, and manage evolution toward the vision. The SE filter also provides an opportunity to more efficiently spend resources and avoid duplication of effort. Managing to SOA requires strong oversight, because growth through satisfying individual interests as they arise is not necessarily the most efficient use of resources.

Some top-down and centralized characteristics of OASIS include the following:

- Centralization played a key role
- Management directives used for migrating applications
- Centralized data center and data management functions
- Centralized release management
- Use of "SOA toolbox"
- Contract vehicles to consolidate IT services contracts
- Close partnership with industry
- Enterprise-wide requirements for architectural control and compliance of all development, hardware and software to an EA reference model and standards
- Service Oriented Architecture governance and use in the context of EA
- Common enterprise data model
- Industry best practices that "extend" the architecture to make the enterprise more adaptable, agile, interoperable, and mission-responsive

3.3.5.2.6 Enterprise Approach

By taking an enterprise-level approach to development, SE helps build in consistency across the NAS and meet NAS-level requirements. This paper and others recommend a business-level (a.k.a. enterprise-level or NAS-level) approach to SWIM, in contrast to an approach focused more narrowly on the technology of SOA or implementation by a few SIPs. There may be a difference between integration of applications within an enterprise vs. across organizational boundaries. Though the basic SOA pattern does not change, the technical strategies and implementation may differ. There is generally more control within a consolidated and centrally managed administrative domain, and a greater degree of predictability about how information will be used. For example, securing information within a single organization may assume a higher level of trust than if information is shared across the organizational boundaries of a federated and independently managed architecture. Application protocols and distributed computing issues are also more important in the cross-business environment.

An enterprise-level approach to development also helps focus on understanding and fixing the right problem. Vendors are differentiating between SOA and Service-oriented enterprise (SOE) to emphasize an enterprise-level approach. The difference between SOA and SOE from a SE perspective is somewhat semantics. When done well, SOA unifies business processes through service-orientation. The distinction is one of scale, specialization, or value. For example, in defining the term "system of systems," it is apparent that one person's system is another's sub-system. Systems-thinking scales to the highest level necessary. Nevertheless, these can be useful paradigms to emphasize focus on solving the right set of problems.

Similarly, some vendors are distinguishing between a "Business SOA" and an "IT SOA." In this paradigm, Business SOA focuses on creating business value through new information products and capabilities, whereas IT SOA focuses more on efficient IT. Shifting the debate away from ESBs, REST vs. SOAP, middleware, and other technologies is not meant to diminish their importance. SWIM depends on a host of networking technologies that work together for scalability. SE cannot design a system without consideration of the underlying technologies. A SOA implementation is still inextricably linked to the technology, though less-so than in traditional IT systems. This dialog reflects a shift in the marketplace, which has solved many of the technical issues, and is moving up the value chain toward selling business solutions, with SOA as a "guiding principle." This is the level where application of SE has its greatest payoff.

More specifically, a business enterprise approach to SE should look at the following:

- Natural patterns of business interactions and behavior
- Current and future interaction with the environment, and the points where it can be influenced
- How information can be used to increase operational efficiency and safety
- How information can be shared at the point of need to improve decision-making
- Anticipate new products, services, and channels for future challenges
- Overall NAS requirements allocated to and supported by SWIM

3.3.5.2.7 System and Organization Patterns Around Boundaries

SOA and SE sometimes require a cultural change within the organization. The reason is tied to the system architecture. In a well-designed system architecture, the boundaries between elements are drawn in areas where little communication is required. When organizing for system development, organization of labor normally occurs early on along the same boundaries as the system architecture. In SOA, this can be a problem when the organization of a business is not service-oriented. A business organization that is customer-oriented, or geographically-oriented is optimized differently than SOA, which lead to challenges in SOA implementation. A successful SOA depends on overcoming these organizational and cultural boundaries. A healthy SOA deployment can be seen in transformation of inter-organizational relationships. This happens in the process of business and IT convergence.

Budgeting and program controls can be difficult in SOA when there is no single organizing model and costs are allocated to a single element for a single purpose, when the very nature of a SOA is built for adaptability to unforeseen circumstances. Benefits may depend on the resources and schedule of other organizations, often beyond the authority and control of a single organization's budget. As budget authority is pushed up higher in an organization, there is less accountability at the lower levels of the organization. Less innovation, and higher risk of single point of failure may result. With higher risk also comes higher reward from a more consistent approach. In this environment, collaboration across traditional organizational boundaries should occur. Though it may seem inefficient, this collaboration is necessary for success. When the concept of SOA takes hold, the only successful implementation is for the organization to start to mirror some of the very principles of the technical architecture.

Here again is a lesson from SE: the boundaries are so important that they can essentially become the point of highest leverage in managing the system. In a SOA, focus should be less on the services as the interactions between them. This is not to say that the services are not important, but less so from a pure architecture perspective.

3.3.5.2.8 Requirements and Verification Best Practices

Requirements can come from anywhere. They may originate with users, and are sometimes hidden as "policies." Completeness is sometimes an issue. To discover what is missing, the level of detail in all areas of the design should be balanced. The weaker areas can be discovered in a number of ways such as through use-case scenarios. Use cases also help identify actors that may have been overlooked and define roles of the actors.

More use cases are better than a few. They provide a template for test and verification. They can also provide an organizational theme for acquisition, such as "buying a use case" or "buying a capability," but are difficult to estimate cost and tradeoff individually. Use cases help provide goals and vision for the architect. They also aid prioritization of the minimum set of things that need to get done to show value. COIs could be engaged in generating cases. And the operations concepts are not only for the architects and leads, but should be widely shared. If an operations concept seems too complex to understand without reading a book, then create as many simplifying themes or use cases as necessary to communicate the concepts.

When managing the design requirements, the rule of thumb is to limit the key system requirements (KSRs) to 7-10 so that a single person is able to keep track of them. This focuses the work of everyone involved by providing a way to monitor and measure performance. Even very large programs can be managed with a handful of KSRs. It is recommended that SWIM come up with their own set of KSRs. They can link directly to metrics of the business functions. System development should be continuously assessed to expected performance using these KSRs.

4 Transition/Migration Approaches for Segment 1 and beyond

This section describes the organizational changes associated with SOA adoption. We will talk about new roles, changed roles, and how the software engineering lifecycle is impacted by SOA.

4.1 How does SOA impact the existing application development methodologies of the enterprise?

SWIM Segment 2 will be the first large scale FAA implementation of SOA. Experience from first Service-Oriented Architecture (SOA) implementation projects suggest that existing development processes and notations such as Object-Oriented Analysis and Design (OOAD), Enterprise Architecture (EA) frameworks, and Business Process Modeling (BPM) only cover part of what is required to support the architectural patterns currently emerging under the SOA umbrella. Thus, there is a need for an enhanced, interdisciplinary service modeling approach[30].

SOA is a modern software engineering design pattern that promises greater mission flexibility and business agility in IT systems. Along with greater flexibility comes greater complexity. SOA emphasizes much more rapid, smaller, incremental changes to existing systems. This will impact traditional software development life-cycle associated with major systems such as NAS, typically characterized by many months, or even years of design, development, testing, and implementation.

SOA adoption should be incremental, with an emphasis on processes. Impact analysis, while practiced extensively in modern software engineering, is emphasized throughout the life-cycle and not simply in regression testing scenarios near the end of a development phase. Proper identification of system components, including components not typically associated with major system deliverables, e.g., design documents, service level agreements, security policies, etc., will require the direct involvement of teams not previously embedded in engineering organizations. These organizational changes will be significant and there is a growing body of knowledge within the government IT community and private industry on how organizations need to change during SOA adoption.

Detailed development models which encompass issues unique to SOA are now readily available and one or more should be adopted as a part of the development of SWIM Segment 2 applications. One example model is “Service-Oriented Modeling and Architecture (SOMA)” developed by IBM. The SOMA mode is described in detail in IBM Systems Journal, artical, "SOMA: A method for developing service-oriented solutions" dated August 6, 2008, http://www.research.ibm.com/journal/sj/473/arsanjani.html.

4.2 What should migrate

REDACTED:
After further review and team discussion, the material envisioned for Section 4.2 on which services would migrate is beyond the scope and visibility of the GEIA Working Group. The Team defers the subject to the FAA to address based on business need and mission requirements. Section 4.5 & 4.6 will address "how" these services should convert into the Shared Services model.

4.3 Scalability of the Solution

In order to insure scalability it is important to avoid a monolithic approach in the design of the SWIM architecture and the NAS systems it supports. As described in Sections 1.2 and 1.3.6, the solution can be partitioned based on service domain, geographic location and service tiers. Complexity can be reduced and scalability increased by federating the SOA functions (Service calls, Governance, ESM, etc) within these partitions.

1 Domain Partitioning - NAS business services are partitioned into independent domains with local infrastructure services that support intra-domain information exchange. This partitioning confines domain specific information exchange to a given domain preventing the central ESB from having to handle all message traffic and enables the domain infrastructure services to be scaled independently from other domains.
2 Geographic Distribution - The Central ESB although managed centrally can be geographically distributed to as many ARTCCs, and even Tracons and airports as needed, to support NextGen incremental evolution and associated increased information sharing.
3 Tiered Services Approach - The tiered services approach enables services within a tier to be scaled independently of other tiers. This scalability applies to any domain implementing a tiered services approach.

In addition, well formed SOA Services are designed to be self contained and as such can be readily distributed locally or geographically to scale at the service level of granularity.

4.4 Test practices and procedures

4.4.1 Testing SOA - Challenges and Approaches

A Complete, Collaborative, and Continuous Approach vs. Traditional Testing Approach

Service Oriented Architecture is a set of design principles and strategies for breaking down legacy, monolithic applications into and introducing new capabilities, as reusable, distributed software components. The result of SOA is increased business agility and visibility, increased information sharing across expanded boundaries of partnerships, cost reductions through re-purposing and leveraging of service components, and unfortunately, <u>a higher degree of infrastructure complexity</u>.

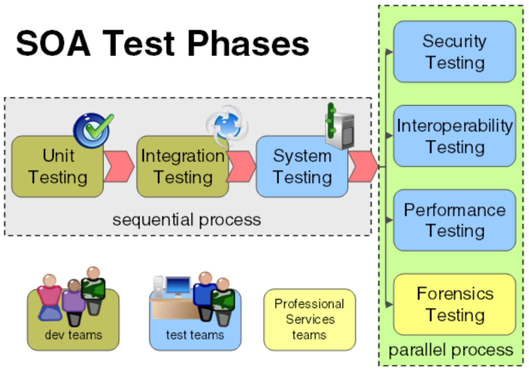
This complexity arises from the fact that with SOA, the "number of moving parts" increases significantly as more and more business logic is allocated to middle-tier services and intermediaries. And these "parts" may be re-used across many different composite services and mission threads, with or without human activities, each requiring different security, transport protocol, data format, and performance requirements, while each having its own unique development lifecycle and dependencies. So SOA, while providing tremendous strategic benefits, also yields increased complexity and higher rates of change to IT.

The increased risk to an organization needs to be mitigated through a testing approach that includes complete, collaborative, and continuous testing, along with methodologies and frameworks that support SOA Testing best practices. Traditional testing approaches are simply inadequate to address the complexity and rates of change to an organizations infrastructure based on SOA.

4.4.1.1 A Complete Approach

Traditional testing for client-server monolithic applications is usually divided between two distinct and independent groups of testers. Development, who often write code for functional unit testing of underlying server logic, and QA, who manually test applications at the User Interface (UI) or "script" automated testing of the UI through the use of COTS.

The approach is analogous to testing a car engine. QA turns the key and the engine does not start, so the test fails. QA informs Development that something is wrong. Unit tests are run on each of the engine parts and record pass tests. What is missing? Tests need to validate the "wiring" aspects of the engine. How do all the parts work together? Of course, this gets even more complicated with SOA when parts are re-purposed and consumed by many composite services or mission threads.



The traditional approach to testing is incomplete for SOA. There are significantly more test phases and layers with SOA that the traditional testing approach does not take into account.

4.4.1.1.1 Challenge: Testing the Many Layers of SOA

With SOA, testing is required at additional layers such as: J2EE (EJB, RMI, Java libraries and API), Web Services (WSDL, SOAP, REST, SAML, WS-I compliance, WS-* standards), Integration Layers (JMS, TIBCO, webMethods, MQ Series, etc.). Tooling should be sought that accommodates testing at all layers of SOA, and for all testing phases of SOA. SOA requires a diagnostic approach to testing.

4.4.1.1.2 Challenge: Writing Code to Test Code

Today, model-driven development is viewed as a necessity to agility. SOA testing requires the same model-driven approach. Writing code or "scripts" to test code is highly inefficient. Model-driven testing provides a pliable, point and click, mechanism for testing all layers of SOA. Tooling which allows point and click introspection and validation of service components, along with parameterization of environment and test data, improves productivity and portability of test assets across multiple testing environments.

4.4.1.1.3 Challenge: Governance and Testing

Testing should be incorporated into a SOA Governance strategy and any testing capability should provide the flexibility and extensibility to interoperate with other governance capabilities, such as, registries/repositories, and enterprise services management. Testing artifacts are themselves SOA artifacts and should be maintained and governed, as such. Traditional testing capabilities are unaware of these critical SOA Governance components and provide little interoperability.

Also, testing should be moved up earlier in the software development lifecycle to the development phase. By early integration testing via simulation/virtualization will demonstrate to the FAA how individual components will interface with each other in production and allowing for the detection and repair of software defects earlier where they are less costly. It also enables developers to simulate end-to-end distributed application scenarios.

4.4.1.1.4 Challenge: Agile Development, Integration, and Testing

Many IT organization seek to become more agile in their development, integration, and testing. Traditional testing approaches require applications be delivered prior to test development. How does testing coincide in parallel to development efforts? Provisions should be made to allow agile testing by permitting parallel development of test assets, even when SOA components are not yet developed or are highly constrained from access. Coding of “dummy” stubs are inadequate for behavioral simulation, and thus service virtualization should be explored as a means for agile development, integration, and testing.

4.4.1.2 A Collaborative Approach

Test early, test often. Testing for SOA needs to be a collaborative approach across development and testing groups. For this to be accomplished, collaborative testing culture must be instilled, and tooling must thread across the roles and functions of these groups, and allow for sharing and re-use of test assets.

4.4.1.2.1 Challenge: Shorten development lifecycle

A collaborative approach to testing SOA will provide higher visibility into failed tests, allowing for component isolation, and faster response.

Traditional testing will allow us to tell the mechanic the engine does not start. SOA testing will allow us to provide a more isolated, descriptive reason as to why the engine will not start, based on a collaborative approach where test assets are being shared and re-used.

4.4.1.2.2 Challenge: Management of Test Assets and Capability

Traditional testing provided different tools and practices for different types of testing i.e. functional, load, performance. Many of these tools are very good at what they do well. But a single tooling that allows complete testing across all test phases promotes agile development. Test assets that are "chained" across test phases are easier to manage and allow reusable test services. In addition, tests assets should be capable of being supported through configuration management systems (CMS).

4.4.1.3 A Continuous Approach

In SOA, services each have their own lifecycle and dependencies. Change occurs at a high rate. Further complicating the issue is that not all services may be under a single organization's control. For these reasons, continuous validation of applications, mission threads, and composite services, needs to be applied, along with proactive alerting of any testing failures, to ensure Quality Assurance at all times.

4.4.2 Solution for FAA SWIM SOA Quality

SOA Testing, Validation and Virtualization solution provides Complete, Collaborative, and Continuous software quality for FAA SWIM.

The FAA SWIM testing solution consists of 3 components:

4 A complete and collaborative **Testing** framework that directly verifies all technology layers with a high level of automation, so teams can better meet defined requirements at lower cost; includes mechanisms for overall governance strategy;
5 Continuous **Validation** so all parties can assure that each software component supports all upstream and downstream dependencies, in an auditable form; and
6 **Virtualization** of the environment, which simulates the behavior and data of the highly interdependent and critical systems the teams must build and test against, even when they may not have access to live or completed components.

4.4.3 The SOA Testing Lifecycle for FAA SWIM

The SOA testing lifecycle for the FAA SWIM program solution mitigates software risks by identifying and resolving problems early in integration and test cycles, and continuously as the services-based system evolves. Here's an overview of the process.

4.4.3.1 Unit tests

Unit tests verify the behavior of software elements. Developers first create unit tests to prove a requirement or demonstrate a bug (they can also receive a test case from a Quality Engineer which exposes the bug). These unit tests should then be reused later as regression tests, or as components or sub-processes of larger workflow tests in Integration and continuously at run-time and change time.

4.4.3.2 Integration testing

Integration testing ensures the various components of a software project operate together. Developers compose their application components to ensure that teams have properly understood their requirements. If some components do not yet exist, Virtual Services can be created to enable testing against a simulated environment even before all the components are complete.

4.4.3.3 System-Level Integration/Testing

System Level Integration Testing consists of a progression of tests designed to verify requirements and identify and mitigate defects throughout the test cycle, reducing risk on delivery of fully functional systems. In addition to receiving the requirements and completed system, Test Engineers can also prove requirements, by taking multiple unit tests and compose end-to-end system tests. This may include expanding boundary conditions and performing deeper data validations. For instance, instead of relying on a code returned from a web service, the test can ensure that database records were inserted, files were created, or ESB messages were sent.

4.4.3.4 Security testing

Security testing evaluation involves the process of discovery and review of all source documentation, scanning and analysis of all operating systems and code, and an evaluation of all identified vulnerabilities. Solution approach complements your network and access security solutions, by allowing your team to build suites of tests that validate not only compliance, but that there are not loopholes in the system logic or interaction between components that would violate intended policies.

4.4.3.5 Interoperability Testing

Interoperability Testing ensures that a new service properly interacts with all the other services. In addition to silo-based testing of individual services, Interoperability Testing makes sure than an entire Mission Thread can be composed and tested as a logical unit, without the need for a User Interface to test. This requires a testing approach which includes invoking the process flow, simulating the activity of real users across the mission, and validating data integrity throughout.

This entire type of integrated testing approach requires testing tools which understand SOA service components such as Web Services (and WS-* standards), Security, XML data validation and Messaging infrastructure. In addition, these tests can run continuously, and interoperate with your SOA Governance and ALM/IT governance management frameworks. Solution flexibility ensures that all teams are working in parallel with their process tools of choice, supported by testing at the needed granularity level for their roles.

4.4.3.6 Performance Testing

Performance testing no longer needs separate teams to script tests and scheduling significant downtime for running tests. Those same test assets which are used to perform functional and end-to-end test cases can be re-run as load tests. This allows Test Engineers the ability to very easily design a working test case, and determine if the scenario still passes properly under load. No additional scripting is required in another tool, the same test assets are re-used across the organization.

Also, if all the components of the System are not available, whether it is because they are under development, or are unavailable or have the wrong data, Virtualization can eliminate that dependency. As soon as 2 components have completed development, an entire environment can be created for isolated testing of those components. As the various services and components become available, the Virtual Services can be very easily replaced with the real implementation. Instead of Test Engineers waiting for numerous environments and systems to be aligned, testing can be run very early in a project lifecycle.

4.5 Facilitating Transition to Segment 2

The dynamic and high demand nature of the NAS demands that it remains in constant operation. Thus, transitioning the NAS to a SOA-enabled environment all at once is not operationally practical, would be prohibitively expensive and is too risky for the underlying mission critical systems with high availability requirements. And, although the FAA may desire evolving to a SWIM-enabled NAS, a full transition to a SOA has implications that may yield economic and organizational considerations.

Important considerations include: a lack of adequate modification time, or rewriting applications for a services environment; legacy component interfaces may be very tightly coupled, making service development and interface definition very difficult; or, it may simply be prohibitively expensive to transition all legacy applications to services, with insufficient return to close a business case.

In addition, some legacy functions may not make suitable services; for example, large data transfers, or functions/services that are called extremely frequently. However, to take advantage of the benefits of SOA, there may be a need to transform some legacy applications into services and provide new capabilities as services. As such, extending existing environments to provide support for services will require an enterprise approach, which has organizational consequences.

The FAA will need to address these cost and organizational constraints in developing an environment capable of executing a hybrid environment, SOA and non-SOA legacy software applications. Hence, the transition to a SWIM-enabled NAS, a cornerstone NextGen enabler, will have to be evolutionary.

4.5.1 Transition Approach

This evolution requires gradually migrating systems to SWIM, introducing stakeholders to SOA over a period of time, with a series of small transitions. The ideal SOA migration strategy will depend on the organization’s motivation for implementing Service Oriented Architecture; hence, the organizational culture and operational paradigm is paramount in migrating to a SWIM-enabled NAS. Nevertheless, the implementation approach is a key aspect in facilitating this evolution of legacy operational systems to SWIM.

As discussed in Section 2.3.6, on-ramping SIPs to a Central ESB, a logical example of a truly Shared Service, through FAA defined standards requires the flexibility to select from a range of interface options for different system components. Figure 15 depicts the NAS as a heterogeneous environment, in which legacy applications and SOA services coexist. The figure illustrates several types of applications interfacing to a Central ESB, via:

- SIPs interfacing directly to the Central ESB (SIP A),
- SIPs leveraging their Segment 1 Service Container (SIP B),
- New NAS Programs with their own implementations of a Program Specific Bus,
- Older NAS Programs with existing SOA implementations requiring custom interfaces, Web Services or Message-oriented-middleware implementations of data exchange to the Central ESB, depending on the capabilities of the SIP.

The Central ESB lends itself to an evolutionary migration. This Shared Services approach offers SIPs the opportunity to participate in NAS-wide information exchange as publishers and/or subscribers, utilizing various SIP tailored, standards-based interfaces, to the Central ESB.

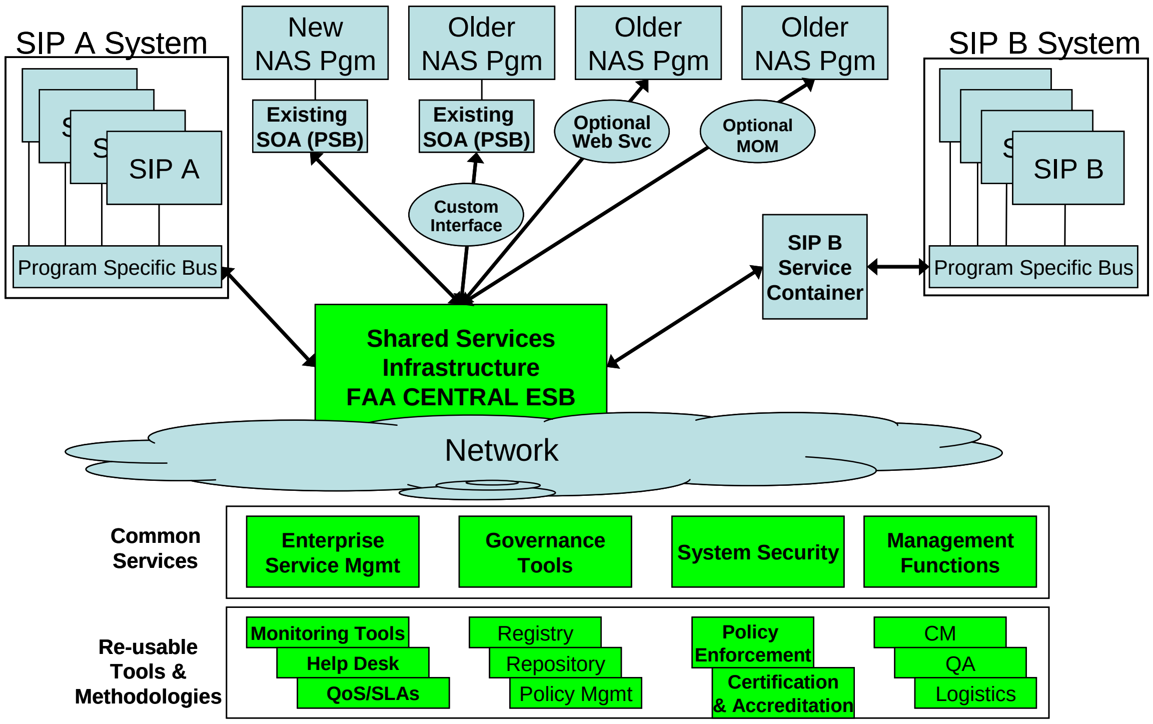


Figure 15: Evolutionary migration requires flexibility

This approach simplifies the FAAs transition to SWIM, by enabling:

- Legacy applications to use their respective legacy infrastructure to support on-going operations, while on-ramping services to a Central ESB,
- Existing SWIM-enabled SIPs to continue their current SOA-based infrastructure to maintain operations,
- The agility for SIPs to rapidly call SOA services as they are made available through the Central ESB, immediately taking advantage of Common Service (ie, registry, etc.) accessible to SIPs,
- New SIPs to on-ramp using tailored, standards-based interfaces, as either producers or consumers.

4.5.2 Critical Success Factors

NextGen success is contingent on the pervasiveness of NAS data availability to SIPs, and SWIM as the enabling technology that ensures this ubiquitous data availability.
To achieve this data ubiquity in their mission, the Defense Information Systems Agency (DISA), through the Net-Centric Enterprise Services (NCES) initiative, has embarked on building the infrastructure to enable the net-centric operations that drive collaboration via an improved information sharing initiative through better access to information, leading to, "superior decision-making across the DoD community" [31]. To achieve this, NCES employs a set of goals referred to as, Net-Centric Data Strategy (NCDS)[32]. The NCDS ensures that data is: (1) visible, (2) accessible, (3) understandable, (4) trustworthy, (5) interoperable, (6) responsive, and (7) institutionalized (meaningful and relevant to mission context).

The FAA can consider these as important goals, which can be benchmarked and considerably facilitate transition. As such, the tools and architecture, processes, and people that support this evolution of data pervasiveness should support a similar set of FAA data strategy goals. Despite the broad spectrum of transition approaches from which the FAA can choose from, the best approach will have to account for NCDS-like goals to maximize data exchange value. The chosen approach will have to balance the requirements of data and service availability, stakeholder responsibilities, operational processes and infrastructure services. The following sections address how these factors can facilitate achieving this evolution, within the context of the NCDS goals referenced above.

4.5.2.1 Operational Processes

The SWIM Segment 2 transition should embrace flexibility as a focal point, accounting for the variability NAS stakeholders participating in this information exchange. Data visibility, accessibility and interoperability are key aspects to ensuring the FAA realizes the benefits that SOAs yield. NAS applications will have to be encouraged and influenced to make data available for NAS-wide use, and utilize data they have never before had access to. Business policies and processes making new data services visible and accessible may need to be developed. Cultural barriers will have to be addressed, guided by governance policies, and balanced against the nature of the participating NAS applications.

For mission critical operations, for example, SOA applications that are introduced must neither destabilize, nor degrade existing mission capability. Mission capability is paramount and the FAA cannot accept any risk that jeopardizes operations or safety. For this reason, it is often desirable to augment existing functionality with new SOA services, instead of replacing legacy functionality. A Shared Services model supports this evolutionary migration.

Current Segment 1 efforts in defining industry COTS-based standards provide a sound foundation for evolving to NAS application interoperability. The FAA can additionally facilitate transition to Segment 2 by establishing operational processes that further promote the visibility and accessibility of data. A critical success factor in operationalizing this will be in ensuring the architecture supports the infrastructure components that enable the commonality of services. The FAAs establishment of an Agency-wide SOA Governance Control Board and SOA Center of Excellence will help ensure a consistent application of policies and processes across NAS applications. These infrastructure services ensure participating NAS programs can, in fact, interoperate and access these data and services, within the institutionalized business policies established by the FAA.

4.5.2.2 Stakeholder Responsibilities

Through this transition, NAS stakeholders will need to clearly understand their roles and responsibilities. Segment 1 SIPs may have defined responsibilities for implementing SWIM Core Services via the Service Container. Hence, the roles of SIPs and their respective interactions with the Shared Services infrastructure must be well defined and understood, within the context of the Segment 1 defined SWIM Core Services. These roles may become somewhat institutionalized and, for a Segment 2 Shared Services model will have to be understood and mapped, accordingly.

For example, SIPs will have responsibility for delivering message content and “data/service packaging” for Enterprise Messaging. Other examples include defining service level agreements for their respective content for Enterprise Service Management, or policy definitions for accessing enterprise service content, as a System Security Core Service. Although some of the tools for executing these roles may be provided as part of the infrastructure services, responsibilities for establishing business policy definitions and operational processes must be clearly delineated.

As suggested in the previous GEIA White Paper on Governance, there are three key areas that need to be addressed within the FAA: (1) SOA Steering Committee; (2) SOA Center of Excellence; and, (3) SOA Governance Control Board. These oversight bodies play a key role in defining these roles and responsibilities.

The SOA Steering Committee, for example, might allocate administrative operational roles to a centralized program office (ie, SWIM Program Office, etc.). Conceivably, this organization would define product specifications (via standards), manage registration and authorize services on-ramping. This team might also grant access to services by producers and consumers, ensuring that applicable policies established by the SOA Governance Control Board are adhered to; such as, messaging, metadata and data content standards. The Steering Committee may further assign the establishment of service level agreements (SLAs) and standardization of metadata, product catalogs.

The SOA Governance Control Board may establish business processes and defining policies for the data exchange. The program team may define interface standards by which NAS applications, monitoring service levels and providing audits for ensuring policy compliance. Many of the tools these various organizations would use to perform their respective operations are can offered as Common Services made available via the Shared Infrastructure Services.

4.5.2.3 Infrastructure Services

The operational processes that NAS stakeholders exercise support their mission and, as such, require infrastructure services that are trusted, responsive, accessible and institutionalized for the mission of the NAS. A Shared Services implementation can provide the requisite components to meet NAS stakeholder requirements for these infrastructure services. The underlying components and infrastructure services must be able to evolve with the migration to

NextGen. Key aspects to this type of transition strategy, then, must be the flexibility of the architecture, scalability of available infrastructure and leveraging existing capabilities.

As illustrated in the Figure 15, Governance Tools, for example, are inclusive of some common service that can be maintained in a logically centralized, physically distributed manner, like registry, repository and policy management. These Shared Service components provide a mechanism for the FAA to administer services, and provide an apparatus for infrastructure component re-use. In the case of enterprise service management (for monitoring services and service levels) and system security (for access control and policy enforcement), for example, enterprise policies can be established into the Shared Services infrastructure and proliferated, accordingly.

Common services are provided via a proliferation of IT infrastructure components that support these Shared Services, like the Central ESB. These IT components are accessed by all NAS application programs that intend to exchange information (content). In a Shared Services model, this common IT infrastructure can be deployed in a manner that will scale to support increasing numbers of NAS stakeholders seeking data and services. This establishes the IT infrastructure as re-usable assets across all participating NAS stakeholders.

Shared infrastructure services promote expansion through incremental investment and reduced long term maintenance costs. The re-use of these IT infrastructure components leverages initial investment and reduces maintenance costs, as they are amortized across a greater number of NAS stakeholders. This type of scalability offers the flexibility of an evolutionary transition to NextGen. Hence, the Shared Services model supports this type of evolutionary transition model.

4.5.2.4 Data and Service Availability

At the root of data and service availability to NAS applications is the visibility and accessibility. To enable considerable in-roads to NextGen, SWIM Segment 2 should provide the capabilities that facilitate that accessibility of data and services. Having the infrastructure services, like a common registry and repository, will promote that visibility and re-usability that enables more pervasive data exchange. Interface standards support the interoperability that promotes a consistent on-ramping of both publishers and subscribers.

The infrastructure services, referenced above, support a proliferation of data, via a scalable infrastructure that achieves these data strategies.

4.5.3 Conclusions

In each case, a Shared Services model provides facilities for a migration to the SWIM-enabled NAS. The use of common services promotes re-use and empowers the FAA with a scalable model that can be funded through incremental investment. These shared services offer the flexibility to on-ramp users by making data and services visible, accessible and interoperable, and a governance mechanism that ensures understandability and trusted enforcement.

For Segment 2, the FAA should consider defining a set of data strategic goals, define the necessary business policies and employ the necessary infrastructure services that maximize re-usability, in a manner that can evolve as the NAS transitions to this new information exchange model. NCDS offers a good benchmark as an FAA starting point in defining a data strategy.

The Shared Service model lends itself to putting the common services in place that address many of the NCDS-like data strategy goals. The critical success factors will require the FAA to address institutional processes and paradigms, address business policies on data availability and stakeholder roles, while ensuring the necessary infrastructure services are available when needed, will all support the evolutionary approach and facilitate a smooth transition from SWIM Segment 1 to Segment 2.

4.6 SOA Transition & Migration Challenges

As described above, there are multiple approaches that can be enlisted to migrate legacy applications to a services oriented architecture. Introducing SOA into operational and maintenance environments presents a number of significant challenges. These challenges span the entire program lifecycle, from procurement through deployment and maintenance. The table below describes some of the most common challenges that can be encountered during SOA migration. For each challenge, a potential solution is outlined. The outlined solution represents one possible solution. There may be other solutions that can result in equally or even more positive outcomes, depending on the specifics of the customer environment.

Factor	*Challenge*	*Solution*
1. Governance	There are typically substantial IT governance capabilities already in place in legacy environments. However, these capabilities must evolve to perform SOA governance.	Basic SOA governance capabilities are critical to the success of the SOA architecture development, migration and operation. A SOA Governance Control Board (SGCB) needs to be established with membership and governance processes defined. The SGCB must ensure that at least basic governance tools are planned for and implemented for managing service lifecycles, developing and monitoring SLAs, and defining and enforcing security policies. Refer to the *SOA Best Practices for FAA SWIM* white paper for additional SOA governance guidance.

Factor	***Challenge***	***Solution***
2. Customer Familiarity with Technology	The various NAS stakeholders and SWIM Implementing Program (SIP) customers may not be familiar with a SOA based approach.	FAA will need to help the stakeholders understand the SOA based approach including development methodologies, migration strategies, associated risks, and expected benefits. FAA SWIM Program objectives, measurable metrics such as reusability goals should be established as early as possible. A SOA Center of Excellence should be established to provide the necessary SOA technical expertise and guidance to support the establishment of the NAS SOA and incremental migration of the legacy applications.
3. Stakeholder Responsibilities	The stakeholders may not clearly understand their own responsibilities related to the NAS migration to a SOA including the associated governance required to ensure success.	Stakeholder roles and responsibilities associated with developing the migration approach, identifying functionality to be exposed as services, and governing the system must be agreed to. Governance of the architecture should start during development and continue throughout the life of the architecture.
4. Contracting Approach	Existing contract and licensing structures may not be compatible with the service based operations paradigm that is enabled via a SOA.	Service level agreements (SLA) will need to be put in place between service providers and service consumers. SLA development and approval must be efficient in order to support quick service deployment. The SLA will need to identify service terms and conditions, service costs, metrics for measuring SLA compliance. The architecture must provide the mechanisms to monitor and enforce SLAs providing metrics necessary to support effect SOA governance.

Factor	***Challenge***	***Solution***
5. SOA Risk Management	SOA development utilizes new methodologies and new technologies introducing different risks than typically experienced with legacy development approaches.	A formalized risk management process should be used to ensure that SOA development and migration risks are proactively identified and mitigation plans developed. Current risk status should be reported to the Governance Control Board for consideration of mitigation plan implementation.
6. Legacy Architecture Documentation	The definition of the current (as-is) NAS architecture may not be documented completely or up-to-date.	Time will need to be allocated to understanding and documenting the as-is architecture. This effort should focus initially on capturing a shallow but broad understanding of the as-is architecture. This effort should then incrementally focus more deeply on the areas of the as-is architecture as they are transitioned as part of the SOA migration effort.
7. Legacy Application and Infrastructure Coupling	Many NAS legacy applications are tightly coupled to the existing infrastructure and other legacy applications. The SOA infrastructure and services that are introduced into the NAS have the potential to destabilize NAS applications and existing infrastructure.	The lowest risk approach is to provide ESB technology, augmented with custom adapters where necessary, to enable interaction with legacy infrastructure or legacy applications during NAS SOA transition and then incrementally upgrade legacy applications to directly utilize SOA technology over time in concert with planned technology refresh cycles. Regression testing of transitioned capabilities must be planned for and performed to ensure no disruption to NAS operations.
8. Effective Service and Information Utilization	SOA services and information needs to be shared among a wide variety of consumers, utilizing diverse technologies, and vendor products.	In order to effectively share services and information, standards selection should begin during the architecture and requirements development phase. However, more than adherence to standards is necessary to ensure interoperability. Common data models and information ontologies need to be developed and shared to ensure information is readily discoverable and usable by consumers.

Factor	Challenge	Solution
9. Performance	Expected performance of SOA COTS infrastructure products in a distributed WAN environment needs to be understood in order to avoid potential performance issues during operations.	Work to identify key performance challenges early, as performance will be a significant architecture driver. Use early prototyping to understand the performance of COTS products and to mitigate performance risks in critical areas.
10. SOA Security	One of the benefits of a SOA is the ability to expose fine-grained capabilities in the form of services to users. As a result, additional security capabilities are needed to protect systems and their exposed services from unauthorized access and potential denial of service attacks.	Additional security policies must be defined. Security policy management and enforcement products are needed to ensure that a SOA does not introduce additional security vulnerabilities and risks into the environment that would impact operations. Specific areas requiring focus regarding a SOA include identify management and access management including both service level security and message level security.
11. SOA Service Availability	Since larger numbers of consumers will be dependent on shared services within a SOA, service availability degradation will have the potential to have more impact. In addition, availability of a domain specific capability may become dependant on a shared service outside of the domain over which the consumer has direct control.	Shared services must be evaluated based on anticipated consumer usage to determine the RMA levels of service required to be supported. In some cases shared services may be required to support multiple RMA levels of service ranging from routine to safety critical. As such shared services may need to be designed to be deployed in multiple environments based the RMA level of service being provided. For example, the service may need to be hosted in a clustered environment, may need to be designed for automated failover, or may need to be replicated. It is necessary to include system RMA requirements in each SLA. When using shared services external to a specific domain, it is essential to ensure that the RMA specified in the SLA for that shared service supports the service RMA requirements of that specific domain utilizing that service.

Table 4: SOA Transition & Migration Challenges

5 Risks

The SWIM Program is conducting and will continue to conduct a Risk Management effort within the SWIM Program To support this activity, a set of Risks are provided in Table 5 below. The intent of this section was to provide the FAA with an initial set of Risks that this team have identified as enterprise level risks that will be associated with the transition and migration to SWIM Segment 2. The risks identified below provide a starter set of risks that the team has identified as relevant to most SOA implementation efforts and will apply to whichever architecture that FAA chooses to implement. This list is by no means complete but is intended to help seed the FAA SWIM Program's Risk management process.

In each risk identified the team has provided a brief description of the risk along with a potential mitigation strategy for that risk. The use of the term "Customer" in the entries in the table is not directed at the FAA or any specific entity but is rather used to represent whatever group or entity is being affected by that specific change. These risks are not unique to the SWIM program but are based on the experiences of this industry working group that appear to be common to all customers that embark on a SOA migration.

Table 5: Enterprise Risks for SWIM Segment 2 Transition

No.	Risk Category	Risk Description	Mitigation Strategy
1	**Standards**	Customer organization has a set of capabilities that have been developed using a specific set of SOA standards, but the standards governing services are immature and evolving which may result in risks due to varying vendor support of standards causing potential schedule and cost impacts. Potential major impact can result if customer infrastructure needs to be upgraded as a result.	Include a task to evaluate existing standards in use, score standards based on compatibility with mission objectives. Monitor commercial and government data sources for standards maturity. Include system controls for gaps in existing standards. Develop a Standards Profile database to track all standards and versions for compatibility. Identify, prioritize and tally industry standards recognizing that conflicts do exist and will impact organizational goals and objectives.
2	**Cost/Schedule**	Customer Organization has high expectations that it will save on initial development costs by adopting SOA but actual SOA development for initial service creation in the short run may be higher versus traditional application development. Service design for a loosely coupled SOA specific application includes establishment of the SOA Governance framework, BPM and system analysis costs as well as the initial SOA learning curve.	Educate organization on SOA fundamentals and cost models. Describe savings in cost and schedule to be gained through reusable, loosely coupled services over the lifetime of the contract. Build a Total Cost of Ownership (TCO) and Return on Investment (ROI) models for SOA services. Clearly delineate the TCO for a SOA infrastructure. Educate customer in other SOA values including flexibility and adaptability which reduces IT budget increases over a rip and replace strategy every 5 years. (src. Key Metrics Can Help Justify Investments in SOA, Michael Barnes, Gartner Group, 2006)
3	**Program Planning**	The Customer has no specific SOA strategy or actionable SOA roadmap making it difficult to plan a transformation to SOA.	The program must actively work with customer to help them define and implement a SOA strategy. Program needs to educate customer in establishing/refining list of imperatives early in the program life cycle and in establishing metrics for monitoring progress towards these goals.
4	**System Status (Manage Service)**	Customer does not have an integrated way to synchronize and manage services and their IT resources.	Mitigate risks in service management by using automated tools to monitor, examine, and analyze the underlying reason for a problem. Use Business Process Modeling (BPM) to examine business impact. Select infrastructure product for visualizing and controlling risks and system status/health.

No.	Risk Category	Risk Description	Mitigation Strategy
5	**System Status (Monitor Service)**	Customer does not have an integrated way to synchronize and manage services and their IT resources.	Mitigate risks in service management by using automated tools to monitor, examine, and analyze the underlying reason for a problem. Use Business Process Modeling (BPM) to examine business impact. Select infrastructure product for visualizing and controlling risks and system status/health, and leverage existing infrastructure management capabilities.
6	**Transition**	Customer organization is a large enterprise distributed across many locations. Extensive risk mitigation needed to ensure smooth transition to SOA enabled ecosystem. Implementation of SOA cannot impact quality or customer services during the transition to new environment.	Section 1.3 was designed to address this risk
7	**Data Integrity**	Customer organization has many subsidiaries and partners with a potential to provide new services or consume services already created. Data integrity and quality are at risk in this scenario and must be effectively managed.	The data that must remain the primary focus of attention for SOA are the data produced, consumed and managed by business systems that represent the past, present or future state of the enterprise. From a business perspective, the concerns are not a matter of distributed storage but how the data are validated, managed and protected. Mechanisms that ensure the data are transported from point A to point B on time, with sufficient resources available, and without performance degradation should be considered. Creating services that "validate, audit, manage, report, and protect" are useful but are not easy to design as they require some technologies and a precise methodology that are relatively new to most IT organizations. Using centralized services for all partners/companies can be helpful.
8	**Governance**	Current business processes automated by IT do not together satisfy our needs and take extended time to develop. Relationship of business to IT is weak and uncooperative.	Incorporate governance, a central registry/repository. Make sure new SOA services perform essential business functions. Involve requestors in the development of new services.

No.	Risk Category	Risk Description	Mitigation Strategy
9	**Governance**	External services consumed by our Customers have the potential to be particularly problematic, uncoordinated, duplicated, and unreliable due to lack of existing governance controls	Crucial to get interfaces and SLAs right prior to implementation and transition to operations. Establish contingency plans in event of unforeseen failures. Requires increased coordination, negotiation (SOA sensitive contracts, documented interfaces, established SLAs), and strong risk management. Educate organization on establishing a mature Governance process to oversee the inclusion of new services.
10	**Governance**	Legacy customer governance policies and mechanisms to effectively manage the definition, development, evolution, and reuse of an organization's service assets can be challenging without some substantive modifications.	Because there is a stronger alignment between IT and business, the SOA governance must address concerns beyond IT governance, including: What decisions must be made to ensure effective, management and use of IT? Who should make these decisions?, How will these decisions be made and monitored? To accomplish these tasks, the SOA Governance should consist of a Governance Organization that will implement and execute governance, a Governance Life Cycle Model for process definition, a Governance Framework to collect and report status, and Policies to enforce rules, processes and behaviors.
11	**Transition**	Migrating legacy systems and operational environments to SOA will typically be a gradual, painstaking process presenting planning/lifecycle challenges.	Document a transition plan that outlines a roadmap and seek approval from the Customer SOA Governance Control Board. Work closely with the Customer to clearly establish expectations, user needs, and roles /responsibilities. Migration can be approached from the infrastructure first, services first, or any combination of the two. Regardless of the approach selected, a strategy to integrate the legacy infrastructure to the SOA infrastructure must be developed. It is helpful to start with a pilot and, upon success, evolve other systems. Also, establish customer objectives, measurable metrics, and reusability goals as early as possible.

No.	Risk Category	Risk Description	Mitigation Strategy
12	**Data Integrity**	Existing organization supports multiple data formats across the enterprise and between companies. No common data dictionary and agreement on common ontology or format of data. Duplicate data collected throughout the organization.	Need to establish a governance framework that oversees data standards, requests for new data, data formats and data retention/archiving policies. Organization needs to agree on common data formats for particular customer community ex (DOD, Justice, Commercial).
13	**Performance**	Customer has introduced new SOA services to legacy systems relying upon existing application performance monitors and test procedures without making adjustments for SOA. Consideration should be given to the applicability of current performance requirements as SOA services may impact system loading, changes in system response time, and changes in service level agreements, routing and security.	Introduction of a SOA validation tool will make it easier to determine system capacity and validate the performance and functionality requirements for a SOA service against a complete set of dependent and distributed applications In a controlled staging environment. The validation tool collects message traffic from the run-time system in order to simulate the system in a staging environment. With the data collected, the validation system checks for impacts from service updates and policy changes to ensure that the SOA system performs and functions as expected prior to production. It serves to mitigate against unforeseen risks prior to going into production.
14	**Performance**	Customer Organization has high expectations for the overall performance of their services which are defined through Service Level Agreements (SLAs). Each customer may have different methods of negotiating SLAs.	One approach is to define the performance parameters that can be used in draft SLAs as early as when the test cases are defined. If the system is using service composition and orchestration, this can affect performance levels, some restructuring may be needed and negotiation can occur as the final SLA is being written. Subject Matter Experts (SMEs) can often help with the resequencing to improve performance. If the customer isn't open to this approach, another consideration is to gather sufficient information about the core processes and gain a good understanding about what kind of performance the customer can live with so good judgment can be applied to orchestration selections that will fall within the ranges of the SLAs.

No.	Risk Category	Risk Description	Mitigation Strategy
15	**Security**	SOA can expose an organization to security risks that can compromise existing network-based internal controls. Web service-enabled networks are more vulnerable to internal and external attacks.	Introduce firewall policies, security controls, standards, and rules that address the particular vulnerabilities of SOA in a production environment. SOA requires programming guidelines Security Assertion Markup Language (SAML), run-time policy enforcement, and governance over changes that may impact system security. Also need to define security extents for SOA chains of calls.
16	**Standards**	Some SOA technologies are becoming mature (e.g., ESBs), but new technologies and standards (e.g., BPM) are continually emerging and many technologies are largely unproven.	Risk management via focused IR&D, industry monitoring, etc., is warranted. Examine existing trade studies for system maturity and flexibility should a technology deficiency develop. Develop a standards profile to identify standards, versions and compatibility for upgrades. Look to COTS vendors for proven roadmaps when upgrading.

6 Recommendations

Table 6 below provides a list of recommendations that this working group believes will help the FAA to move forward in defining the details of the Segment 2 architecture and the resulting requirements as well as provide some key mechanisms for SWIM program management. Some of these recommendations are high level and will spawn multiple actions to provide the more specific and detailed studies, analyses and work products that are needed to successfully implement SWIM across the NAS.

As with the Risks identified in Section 5 above, this list of recommendations is not intended to be a definitive or all inclusive list of recommendations but rather an initial list designed to provoke more thought and analysis within the FAA SWIM team.

No.	Recommendation	Reference
1	FAA should conduct key studies to provide framing data to the problem space of SWIM through federated and consolidated services in order to obtain useful and pragmatic solutions.	2.3.3 – Relevant Issues Associated with Alternatives
2	A trade study be conducted to determine the exact order of importance of each of the SOA architecture selection criteria are to SWIM 2.	2.3.4 - Selection Criteria and Justification
3	The FAA should consider defining a set of data strategic goals, the necessary business policies and employ the necessary infrastructure services to maximize re-usability, in a manner that can evolve as the NAS transitions to Segment 2.	4.5.3 - Facilitating Transition to Segment 2 – Conclusion
4	FAA should consider developing a "net-centric data strategy", NCDS-like, goals for establishing the SOA.	4.5.3 - Facilitating Transition to Segment 2 – Conclusion
5	FAA should pursue activities that would establish re-usable infrastructure services where any SIPs (Segment 1 & Segment 2) can begin to leverage for commonly used services.	4.5.3 - Facilitating Transition to Segment 2 – Conclusion
6	Establishment of a SOA Steering Committee, SOA Governance Control Board & SOA Center of Excellence.	4.5.2.2 - Facilitating Transition to Segment 2 – Stakeholder Responsibilities
7	FAA should establish the mechanism(s) to ensure the Segment 2 architecture and beyond provides for the mechanisms to monitor and enforce SLAs providing metrics necessary to support effective SOA governance.	4.6 – SOA Transition & Migration Challenges

No.	Recommendation	Reference
8	FAA should ensure as part of the SWIM Risk Management process , that specific SOA development and migration risks are identified and risk status reported to the Governance Control Board for consideration of mitigation plan implementation.	4.6 – SOA Transition & Migration Challenges
9	FAA should use prototyping to conduct performance analysis of the SOA COTS infrastructure in a distributed WAN environment to understand and avoid potential performance problems during operations.	4.6 – SOA Transition & Migration Challenges

Table 6:   Recommendations for FAA Consideration

7 Appendices

7.1 FAA SWIM Acronyms

Below is a list of key acronyms the FAA uses to discuss operations and the environment SWIM will support. Not all of these terms arise in the current whitepaper, but they appear here for completeness and to support future discussions.

ADDS	Aviation Digital Data Service
ADOC	Airline Direct Operating Cost
AFTN	Aeronautical Fixed Telecommunications Network
AIM	Aeronautical Information Management
AOC	Airline Operating Center
ARMT	Airport Resource Management Tool
ARTCC	Air Route Traffic Control Center
AS	Application Server
ASDE-X	Airport Surface Detection Equipment – Model X
ATC	Air Traffic Control
ATCT	Air Traffic Control Tower
ATO	Air Traffic Operations
AWC	Aviation Weather Center
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPM	Business Process Management
C-ATM	Collaborative Air Traffic Management
CA	Certificate Authority
CBR	Content Based Routing
CDM	Collaborative Decision Making
CERAP	Center Radar Approach Control
CIWS	Corridor Integrated Weather System
CMP	Configuration Management Plan
COI	Community of Interest
CORBA	Common Object Request Broker Architecture
COTS	Commercial off-the-Shelf
CP	Central Processor
CPMP	Commercial Product Management Plan
CSIRC	Computer Security Incident Response Center
DAFIF	Digital Aeronautical Flight Information File
DNS	Domain Name Service
DOTS	Dynamic Ocean Track System
EAP	Extensible Authentication Protocol
EFSTS	Electronic Flight Strip Terminal System
ERAM	En Route Automation Modernization
ESB	Enterprise Service Bus
ESM	Enterprise Service Management
ESP	Encapsulating Security Payload

ETE	End-to-end
EVM	Earned Value Management
FAA	Federal Aviation Administration
FBWTG	FAA Bulk Weather Telecommunication Gateway
FDIO	Flight Data Input Output
FEA	Federal Enterprise Architecture
FID	Final Investment Decision
FPR	Final Program Requirements
FSS	Flight Service Stations
FTI	FAA Telecommunications Infrastructure
FTP	File Transfer Protocol
FY	Fiscal Year
GCNSS	Global Communications, Navigation, and Surveillance System
HADDS	Host Automation Data Distribution System
HIDS	Host-based Intrusion Detection Sensor
HT	Hypertext and Transfer Protocol
ICAO	International Civil Aviation Organization
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
ILS	Integrated Logistics Support
ILSP	Integrated Logistics Support Plan
IOC	Initial Operating Capability
IOT&E	Independent Operational Test and Evaluation
IOTRD	Independent Operation Test Readiness Decision
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
IPS	Internet Protocol Service
IPSec	IP Security
ISD	In-Service Decision
ISR	In-Service Review
ISS	Information Systems Security
IT	Information Technology
ITWS	Integrated Terminal Weather System
J2EE	Java 2 Platform, Enterprise Edition
JMS	Java Messaging Service
JPDO	Joint Planning and Development Organization
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LOA	Letter of Agreeement
MADE	Military Airspace Data Entry System
MIME	Multi Purpose Internet Mail Extensions
MOM	Message oriented Middleware
MOU	Memorandum of Understanding
MQ	Message Queuing
MTOM	Message Transmission Optimization Mechanism
NACO	National Aeronautical Cartographic Organization
NAIMES	NAS Aeronautical Information Management Enterprise System
NAS	National Airspace System
NASE	NAS Adaptation Services Environment

NASR	National Airspace System Resources
NextGen	Next Generation Air Transportation System
NGATS	Next Generation Air Transportation System
NIDS	Network Intrusion Detection Sensor
NIST	National Institute of Standards and Technology
O&M	Operations and Maintenance
OPSCON	Operations Concept
OT&E	Operational Test and Evaluation
PDC	Pre-Departure Clearance
PDR	Preliminary Design Review
PHA	Preliminary Hazard Analysis
PIREP	Pilot Report
PKI	Public Key Infrastructure
POC	Point of Contact
PP	Protection Profile
PSB	Program Specific Bus
PTR	Program Trouble Report
PVT	Passenger Value of Time
QAP	Quality Assurance Plan
QoS	Quality of Service
RFC	Request for Comment
RMI	Remote Method Invocation
RVR	Runway Visual Range
SA	SWIM Adapter
SAML	Security Authorization Markup Language
SAMS	Special Use Airspace Management System
SD	Situation Display
SDP	Service Delivery Point
SEC	Systems Engineering Council
SIG	Security Incident Group
SIP	SWIM Implementing Program
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOW	Statement of Work
SNMP	Simple Network Management Protocol
SRVT	Safety Requirements Verification Table
SMS	Safety Management System
SRMGA	Safety Risk Management Guidance for Acquisitions
SSAR	System Safety Assessment Report
SSD	System Specification Document
SSH	System Safety Handbook
SUA	Special Use Airspace
SWIM	System Wide Information Management
TBD	To Be Determined
TCP	Transmission Control Protocol
TDDS	Terminal Data Distribution System
TDLS	Terminal Data Link System

TFM	Traffic Flow Management
TFM-M	Traffic Flow Management – Modernization Program
TFMS	Traffic Flow Management System
TRACON	Terminal Radar Approach Control
TSD	Traffic Situation Display
TSG	Telecommunications Service Group
UDDI	Universal Description, Discovery, and Integration
UDP	User Datagram Protocol
URI	Uniform Resource Indicator
URL	Uniform Resource Locator
USNS	United States Notice to Airmen (NOTAM) System
VNTSC	Volpe National Transportation Systems Center
VPN	Virtual Private Network
VRTM	Verification Requirements Traceability Matrix
WAN	Wide Area Network
WARP	Weather and Radar Processor
WINS	Weather Information Network Server
WJHTC	William J Hughes Technical Center
WMSCR	Weather Message Switching Center Replacement
WSDL	Web Services Definition Language

7.2 Foot Notes

---

[1] ITAA-GEIA, FAA SWIM Program SOA Best Practices White Paper, version 7.0 dated March 24, 2008.
[2] Joint Planning and Development Office, *Next Generation Air Transportation System 2005 Progress Report*, March 2006. See also: Reference to JPDO Eight (8) Key Capabilities: (1) Net-Enabled Information Access
(2) Performance-Based Services, (3) Weather-Assimilated Decision Making, (4) Layered, Adaptive Security,
(5) Position, Navigation, and Timing Services, (6) Trajectory-Based Aircraft Operations, (7) "Equivalent Visual" Operations, (8) "Super Density" Operations
http://www.jpdo.gov/library/20070726AllHands/20070727_JPDOAllHandsMeeting_JPDOProgressandOutlook_Leader_FINAL.pdf, Page 3

[3] System Wide Information Management (SWIM) Technical Overview, Version 1.1, Page ES-3 & Section 5. Updated: 28 March, 2008

[4] System Wide Information Management (SWIM) Technical Overview, Version 1.1, Page ES-4 – ES-5. Updated: 28 March, 2008.

[5] "System Wide Information Management (SWIM) Service Container Requirements", Page 2, FAA. Nov 2007.

[6] "Concept of Operations for the Next Generation Air Transportation System", Version 2. June 13, 2007.

[7] Gartner, **Open Source in MOM**, March 2008, G00156079

[8] All trademarks, service marks, copyrights, and other rights are the property of their respective owners.

[9] Second most widely downloaded opensource JMS MOM [Gartner (March 2008)]

[10] Most widely downloaded opensource JMS MOM [Gartner (March 2008)]

[11] Gartner, **Open Source in MOM**, March 2008, G00156079

[12] All trademarks, service marks, copyrights, and other rights are the property of their respective owners.

[13] Does not use JMS

[14] Does not use JMS

[15] Most widely-used closedsource JMS MOM [Gartner (March 2008)]

[16] Does not use JMS

[17] Gartner, **Open Source in MOM**, March 2008, G00156079

[18] XML Infoset is a World Wide Web Consortium (W3C) specification describing a data model of an XML document in terms of a set of information items.

[19] Assuming UTF-8 text encoding

[20] Send and receive files using a combination of SOAP and MIME

[21] W3C recommended convention defined for efficient serialization of XML Infosets that have a mix of binary and textual data

[22] A Practical Guide to Federal Service Oriented Architecture Version 1.1, http://smw.osera.gov/pgfsoa/index.php/Version1.1#Establish_Federated_Governance

[23] ITAA-GEIA, FAA SWIM Program SOA Governance Best Practices Industry Input September White Paper, version 1.0 dated 24 September 2008.

[24] SOA Governance Best Practices Industry Input, p. 24, September 2008

[25] Mark Bouchard, CISSP,- Missing Link Security Services – Unknown Attacks Paper for Secure Computing

[26] Symantec Internet Security Threat Report, Trends of January 05 – June 05, Volume VIII, Published September 2005

[27] Symantec Internet Security Threat Report, Trends of January 05 – June 05, Volume VIII, Published September 2005

[28] Symantect Internet Security Threat Report, Trends of January 05 – June 05, Volume VIII, PublishedSeptember 2005

[29] Marcus J. Ranum - expert on security system design and implementation – Dude! Paper for Secure Computing.

[30] Clive Gee (clive@us.ibm.com), Senior Solution Architect, IBM, Pal Krogdahl (pal.krogdahl@se.ibm.com), Solution Architect, IBM, Olaf Zimmermann (ozimmer@de.ibm.com), Senior IT Architect, IBM "Elements of Service-Oriented Analysis and Design: An interdisciplinary modeling approach for SOA projects" (02 Jun 2004) http://www.ibm.com/developerworks/webservices/library/ws-soad1/

[31] http://www.disa.mil/nces/

[32] http://metadata.dod.mil/mdr/ns/ces/techguide/net_centric_data_strategy_ncds_goals.html